Leakage-Resilient Cryptography with Key Derived from Sensitive Data

No Author Given

No Institute Given

Abstract. In this paper we address the problem of large space consumption for protocols in the Bounded Retrieval Model (BRM), which require users to store large secret keys subject to adversarial leakage. We propose a method to derive keys for such protocols on-the-fly from weakly random private data (like text documents or photos, users keep on their disks anyway for non-cryptographic purposes) in such a way that no extra storage is needed. We prove that any leakage-resilient protocol (belonging to a certain, arguably quite broad class) when run with a key obtained this way retains a similar level of security as the original protocol had. Additionally, we guarantee privacy of the data the actual keys are derived from. That is, an adversary can hardly gain any knowledge about the private data except that he could otherwise obtain via leakage. Our reduction works in the Random Oracle model.

As an important tool in the proof we use a newly established bound for min-entropy, which can be of independent interest. It may be viewed as an analogue of the chain rule – a weaker form of the well-known formula $\mathbf{H}(X|Y) = \mathbf{H}(X,Y) - \mathbf{H}(Y)$ for random variables X, Y, and Shannon entropy, which our result originates from. For min-entropy only a much more limited version of this relation is known to hold. Namely, the min-entropy of X may decrease by up to the bitlength of Y when X is conditioned on Y, in short: $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X) - |Y|$. In many cases this inequality does not offer tight bounds, and such significant entropy loss makes it inadequate for our particular application. In the quasi chain rule we propose, we inject some carefully crafted side information (spoiling knowledge) to show that with large probability the average min-entropy of X conditioned on both: Y and this side information can be almost lower bounded by the min-entropy of (X,Y) decreased by the min-entropy of Y conditioned on the side information.

1 Introduction

1.1 Key derivation from sensitive data

In this paper we make an attempt to adapt the problem of overcoming weak expectations, which recently has attracted considerable attention in the cryptographic community [2,11,28], to yet another well-recognized setting – the Bounded Retrieval Model (BRM) [7, 12]. Here, the term *weak expectations* refers to the (expected) chances of breaking a cryptosystem when an imperfect source of randomness is employed in places where uniformly random bits were supposed to be used. For instance, one can quantify security of a system with semi-random keys used instead of keys drawn from uniform distribution. What motivates such analysis is that the standard assumption about unlimited availability of truly random bits turns out to be overoptimistic in practice. On the other hand, cheap sources of weak randomness can be easily found "in nature". Suffice it to mention physical sources or biometric data [4,9], which remain somewhat unpredictable for adversaries.

In the conventional approach to cryptography, security of a scheme relies on privacy of cryptographic keys. The dawn of so called *side channel attacks* has influenced this perspective significantly. There, an adversary may gain some partial knowledge about the secret keys, e.g., by measuring timings [19], power consumption [20], electromagnetic radiation [24], or even sounds (acoustic cryptoanalysis) [15] emitted by a device a cryptographic protocol is implemented on. Leakage-resilient cryptosystems are meant to address attacks of this form and remain secure when the adversary is allowed to adaptively learn arbitrary functions of the secret keys subject to only one restriction – namely, the total length of information leaked in the process must not exceed a leakage bound λ. There are in fact two slightly different models of leakage considered in the literature. The *relative leakage* allows λ to be some fraction of the length of a secret key. In the *absolute leakage* setting, also known as the BRM [14], the parameter λ is fixed in the first place, and then the length of a key may be chosen accordingly, depending on λ, to achieve the desired level of security. It is important to note that this flexibility in increasing the key size does not affect other parameters possibly present in a BRM protocol, such as computation or communication complexity – these should only depend on a security parameter but not on λ.

Space-efficient BRM The leakage bound λ and, consequently, the size of a key in the BRM are typically very large, the latter being of order of gigabytes. Although per-gigabyte storage cost is becoming lower every year, this downside of BRM protocols may still be an issue, e.g., for many mobile devices with quite limited size of non-volatile memory available. When combined with the fact that keys used in these protocols are required to be sufficiently random, it means that computers running a BRM protocol are clogged with some huge blob of random and otherwise useless data. In the solution we propose, BRM keys can be derived on-the-fly (that is, it is not necessary to keep them on disk, and they may be computed when a relevant portion of the key is requested) from data a user want to store on his disk for any other reason. The private user data usable in this context may include: text documents, photos, audio files, or other media. This may lessen the problem of wasted disk space however for a reduced space we trade in additional computations needed to determine BRM keys.

An issue that arises here is that such data, when viewed as a source of randomness, while being unpredictable, to a degree, for an adversary, is certainly not uniformly random (e.g., note that certain segments in some file formats may be fixed or come from a prescribed set of values). Here, the connection to the aforementioned problem of overcoming weak expectations and, which is related, key derivation, becomes apparent.

Overcoming weak expectations A study of cryptographic applications that retain a comparable level of security when fed with weakly random sources instead of ones having uniform distributions was initiated by Barak *et al.* [2]. There, the authors explore the idea of applying universal hash functions to key derivation. The renowned Leftover Hash Lemma (LHL) [17] states that families of such functions constitute good randomness extractors. Specifically, when applied to a source of min-entropy k, an extractor of this form produces m bits which are δ-close (in terms of statistical distance) to uniform, as long as $k \geq m + 2\log(1/\delta)$. A key obtained this way can be then used in a cryptographic application. The min-entropy loss of magnitude $2\log(1/\delta)$ may be unacceptably large in some situations but, as shown by Radhakrishnan and Ta-Shma [25], it cannot be prevented in general. However, as argued by the authors, there exists a wide range of applications where the entropy loss can be cut down by the factor of 2 for a price of some security loss in the application using non-uniform keys. This line of research was continued by Dodis and Yu [11].

Sensitive data Building a cryptographic protocol on top of randomness derived from private data bears an obvious risk of compromising that data. One can imagine an artificial protocol that simply publishes all accessible randomness. Also, a protocol in the BRM does not necessarily guarantee protection of its key. Some fragments of a BRM key may be passed, as a part of normal operating procedure, to an honest party that did not possess the key in the first place. To give an example illustrating such a situation, one can conceive of an authentication protocol in the BRM, which itself appears to be folklore, based on Merkle tree [21]. There, a hash tree is built on an input BRM key and the resulting hash from the root is then forwarded to a verifier (say, a bank). This way a user can commit to his key which, in its entirety, is only stored on user's side for efficiency reasons. On the other hand, the verifier may learn parts of the key when the user attempts to authenticate himself. In order to do that, the verifier demands to present hashes along some path of his choice in the Merkle tree. Such a path includes data from the initial BRM key and thus its fragment gets revealed to the verifier.

Now, if a BRM key used in this protocol is obtained from data stored on disk then, clearly, the key derivation procedure should enjoy some kind of a one-wayness property. If the procedure does not hide its input then a dishonest verifier may attempt to recover the underlying data or, at least, he may gain some partial knowledge. In this paper, we aim at a solution that allows a user to protect his private and possibly sensitive data in this scenario. Namely, we require that an adversary can hardly learn anything more about the data except that he could otherwise achieve via leakage.

Overview of our solution Seemingly, the problem of extracting an almost random key from sufficiently random data can be easily solved, even in presence of leakage, using a well-known primitive – namely, an average-case strong randomness extractor. Its definition requires that for any two random variables X and I (where I can be viewed as side information about I, i.e., a leak) such that the (conditional) min-entropy of X given I (see (1) for a precise definition of conditional min-entropy) is high enough, then the output of the extractor $\mathrm{Ext}(X, R)$ is statistically close to uniform even given a short random seed R and the side information I, in short: $\big(\mathrm{Ext}(\mathrm{X}, \mathrm{R}), R, I\big) \approx (U, R, I)$. Dodis *et al.* [9] extend the LHL to show that universal hash functions constitute good aveon-rage-case extractors retaining nearly the same parameters as in the original LHL. We also note that the definition of such extractors is enough to cover the privacy requirement in our particular application – if $\mathrm{Ext}(X, R)$ disclosed some information about private data X then by setting I to be this information we would produce a correlation between $\mathrm{Ext}(X, R)$ and I, thus violating the condition about $\mathrm{Ext}(X, R)$ being close to uniform and independent of I. Overall, randomness extractors allows us to cover the two main properties we aim at in this paper, i.e., the uniformity of keys and the privacy of underlying data. However, such a construction would be downright impractical. From the computational point of view, extractors are not suited best to work on inputs as huge as in our application. Also, they are inherently non-local in the sense that each bit of an output should depend on almost every bit of an input. This means that in order to compute even a small portion of the derived key on demand using an extractor, one has to read and supply almost the whole input data which is not a viable option.

To address the issue related to efficiency and locality, we propose a different way of deriving keys from private data. Our idea is quite straightforward – it boils down to splitting all the data into consecutive blocks of the same fixed length n (say, $n = 4\mathrm{kB}$). A block could naturally correspond to the smallest allocation unit in a filesystem present on a user's device. Then, we use hashing to extract randomness from blocks. The naïve method to implement it would be computing hashes block by block. This approach, albeit simple, has a significant drawback. The only assumption we make about the input data is that its joint min-entropy is not too small (this measures the *a priori* knowledge of the adversary about the private data, before leakage is taken into account). We do not demand however that the randomness is equidistributed across all the blocks. Therefore, it may happen that even for high overall min-entropy, e.g., $\frac{1}{2}\ell n$ where ℓ is the number of blocks, there exist $\ell/2$ blocks which, from the adversary's point of view, are constant. Consequently, the corresponding parts of a derived key carry no randomness at all and are known to the adversary.

We circumvent the problem caused by blocks with low min-entropy we increase the number of blocks a single block of a derived key depends on. That is, each hash is calculated by taking not one but d blocks of input. Additionally, we amplify the likelihood of the event that there is at least one high min-entropy block among the selected d-tuple. This step actually introduces a new flavor to the reasoning. Namely, we argue the assumption on joint min-entropy of the input blocks with large probability implies that there exists a large number of blocks each having high min-entropy. This statement may seem rather natural and intuitive yet it is somewhat tricky to prove. A related problem of extracting random blocks was considered before by Nisan and Zuckerman [23] and Alwen, Dodis, and Wichs [1].

The fact that there should be plenty of sufficiently random blocks in the input allows us to pick d-tuples of block randomly. However, to recreate portions of the derived on-the-fly one would have to store the auxiliary randomness used to select those tuples, which may not be acceptable. Instead, we suggest employing dispersers – d-regular bipartite graphs with the property that any sufficiently large set of vertices on the left side is connected to almost all vertices on the right side. Every such a disperser induces a selection of d-tuples.

Clearly, increasing the degree of regularity d of a disperser reduces locality of the key derivation method. This however comes as a trade-off. We use a simulation-based argument to prove that any protocol using the derived key can be simulated by a protocol operating on an original key with $O(n\ell/d)$ of additional leakage.

1.2 Chain rule for min-entropy

Since the beginning of formal treatment of cryptography most works have heavily relied on different flavours of *entropy*. Depending on the context, those notions are used to measure compressibility, unpredictability or uncertainty of outcomes of random processes. In his seminal work Shannon applied the simplest, compressibility notion of entropy (called after his name), i.e., the one defined by $\mathbf{H}(X) \stackrel{\text{def}}{=} \mathbb{E}_x \log \frac{1}{\Pr(X=x)}$ to prove that in a perfectly secure symmetric key encryption scheme the length of the key is necessarily as large as the length of the message. The notion which turned out to be even more useful in the area of cryptography is *min-entropy* defined by the formula $\mathbf{H}_\infty(X) \stackrel{\text{def}}{=} -\log(\max_x \Pr(X=x))$ and encompassing unpredictability properties of a random variable X.

Conditional entropy Shannon's entropy possesses a natural generalization to its conditional version $\mathbf{H}(X|Y)$ which satisfies the formula $\mathbf{H}(X,Y) = \mathbf{H}(X|Y) + \mathbf{H}(Y)$. This corresponds to an intuitive interpretation stating that the information contained in (X,Y) consists of the information in Y extended by the conditional information in X given Y. Dodis *et al.* [9] provided an analogous notion for min-entropy. Namely, for two random variables X, Y the conditional min-entropy $\widetilde{\mathbf{H}}_\infty(X|Y)$ is given by the formula:

$$\widetilde{\mathbf{H}}_\infty(X|Y) \stackrel{\text{def}}{=} -\log\left(\mathbb{E}_y 2^{-\mathbf{H}_\infty(X|Y=y)}\right). \tag{1}$$

This definition turns out to preserve the natural interpretation of min-entropy as maximal probability of success in guessing X given Y, i.e., for any algorithm $\mathcal{A}$ we have:

$$\Pr(\mathcal{A}(Y)=X) = \mathbb{E}_y \Pr(\mathcal{A}(y)=X) \leq \mathbb{E}_y 2^{-\mathbf{H}_\infty(X|Y=y)} = 2^{-\widetilde{\mathbf{H}}_\infty(X|Y)}.$$

Regrettably, the above definition possesses serious drawbacks explained in the following example.

Example 1 (Cross distribution). Let $X = (X_1, X_2) \in (\{0,1\}^n)^2$ be a random variable distributed uniformly over "the cross", i.e., a set $\{0,1\}^n \times e \cup e \times \{0,1\}^n$ for some fixed $e \in \{0,1\}^n$. Note that, we have $\mathbf{H}_\infty(X_1, X_2) = -\log \frac{1}{2^{n+1}-1} \in [n, n+1]$ and $\mathbf{H}_\infty(X_i) = -\log \frac{2^n}{2^{n+1}-1} < 1$ and therefore, the sum property does not hold

without any further assumptions or conditions. Moreover, $\widetilde{\mathbf{H}}_\infty(X_2|X_1) < \mathbf{H}_\infty(X_2)$ and therefore $\widetilde{\mathbf{H}}_\infty(X_2|X_1) + \mathbf{H}_\infty(X_1) < 2$ which consequently means that the most natural chain rule does not hold either.

The authors also prove the following result.

Lemma 1 (Lemma 2.2 in [9]). *Let X, Y, Z be random variables. Then*

a) For eny $\delta > 0$, the conditional entropy $\mathbf{H}_\infty(X|Y = y)$ is at least $\widetilde{\mathbf{H}}_\infty(X|Y) - \log(1/\delta)$ with probability at least $1 - \delta$ over the choice of y.

b) If Y has at most 2^λ possible values, then $\widetilde{\mathbf{H}}_\infty(X|(Y, Z)) \geq \widetilde{\mathbf{H}}_\infty((X, Y)|Z) - \lambda \geq \widetilde{\mathbf{H}}_\infty(X|Z) - \lambda$. In particular, $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X, Y) - \lambda \geq \mathbf{H}_\infty(X) - \lambda$.

The above item b) of Lemma 1 is treated as chain rule for min-entropy. Its significant weakness is that the inequality does not depend on the random properties of Y but its actual size λ. We illustrate this by a simple example.

Example 2 (Two blocks almost half entropy). Let X, Y be two random variables distributed over $\{0,1\}^n$ with joint distribution of min-entropy $\mathbf{H}_\infty(X, Y) = n$. Then, the above item b) gives us a trivial estimate $\widetilde{\mathbf{H}}_\infty(X|Y) \geq \mathbf{H}_\infty(X, Y) - |Y| = 0$.

Nevertheless, if we condition (X_1, X_2) given in Example 1 with a random variable Hint defined by the formula

$$\mathrm{Hint} = i \iff X_i = e,$$

then the second variable X_{3-i} has conditional min-entropy $\mathbf{H}_\infty(X_{3-i}|\mathrm{Hint} = i) = n$. Therefore, there exists a certain additional "knowledge" Hint which allows us to extract almost whole min-entropy from the pair (X_1, X_2). Namely, the event

$$\mathbf{H}_\infty(X_1|\mathrm{Hint} = i) + \mathbf{H}_\infty(X_2|\mathrm{Hint} = i) \geq \mathbf{H}_\infty(X_1, X_2) - 1$$

holds with probability 1 over the choice of i. This suggests that the right way to obtain the chain rule is additional conditioning. Similar approach was used in certain different applications and is classically called *spoiling knowledge*.

Remark 1. We believe that our results might find a meaningful application in the theory of extractors as we exhibit a certain (non-strict) block-wise structure of any distribution of high min-entropy (cf. proof of the Claim inside Corollary 2). Block-wise distributions are widely used in theory of extractors (see, e.g., [22]).

Previous results concerning chain rule. Previous research concerning (or somehow related) to chain rule is not only mentioned in discussed above [9]. For example authors of [1] and [23] prove that random (sufficiently large) subtuple of some set of variables with high min-entropy must preserve some significant amount of this entropy. Our result can be viewed as a generalization of this fact since from our reasoning we get that some *specific* (not random) subtuple preserves some min-entropy. (see Corollary 2.) Moreover in [23] authors try to deal with the problem of chain rule for min-entropy but need to make big effort to get some complicated workaround since they do not have any quasi chain rule for min-entropy in hand.

Moreover, they show a simplified version of their result and give a short brief proof based on chain rule for Shannon entropy.

Another important previous result is Lemma A.1 (min-entropy split) from [5] (and also other variants from [8, 27]). Here authors formulate a theorem that also can be viewed as a quasi chain rule. As an example compare with the following:

Lemma 2 (Lemma 4.2 (Min-Entropy-Splitting Lemma) in [8]). *Let $\epsilon \geq 0$ and let X_0, X_1 be random variables (over possibly different alphabets) with $\mathbf{H}^{\epsilon}_{\infty}(X_0X_1) \geq \alpha$. Then, there exists a binary random variable C over $\{0,1\}$ such that $\mathbf{H}^{\epsilon}_{\infty}(X_{1-C}C) \geq \alpha/2$.*

This is very interesting result that shows that it is possible to extract partial min-entropy from a pair of variables. However authors justify high min-entropy of just one viariable from the pair. In our result we get significantly more dealing with both variables at once. See Lemma 3 for details.

2 Our contribution

The results of the paper are twofold.

Making BRM space efficient. In this paper, we give a new idea to overcome a problem with large space requirements in the BRM model. As a reminder: BRM uses huge private keys for purpose of leakage-resiliency. Here we describe an idea to derive secret key from private data (this could include text documents, videos, etc.). That content is supposed to have high enough min-entropy, however it raises a problem with privacy: we do not want to reveal any sensitive data outside. Our construction fulfills this expectation. So the private data remains undisclosed even if the entire derived key is compromised.

The secret key is being computed on-the-fly from private data so that no extra memory is used to store the key. Access to the key is fast so one does need to read limited portion of private data to compute some part of the secret key.

The main result shows that any cryptographic protocol from well defined and vast class (intuitively: game based protocols) is still secure if we use a key derived from private data in place of a random key.

Chain rule through spoiling knowledge. The reasoning from Section 1.2 exhibits a pair of random variables X, E such that $\mathbf{H}_{\infty}(X|E=e) + \mathbf{H}_{\infty}(Y|E=e) \gg \mathbf{H}_{\infty}(X) + \mathbf{H}_{\infty}(Y)$ with probability 1 over the choice of e. Analogous simplified situations were investigated by Bennett *et al.* [3] for collision entropy $\mathbf{H}_2$ and utilized in privacy amplification. Furthermore, similar examples also exists for other Renyi entropies $\mathbf{H}_{\alpha}$ for $\alpha > 1$ and were systematically analysed by Cachin in [6] in context of smooth entropy. Our methods substantially generalize "profiling" method of Cachin and Maurer and gives a precise spoiling knowledge sufficient to obtain chain rule for min-entropy. Our main result in this part of the paper is the following (see Lemma 3):

Chain rule. Let X, Y be random variables. Then, there exists a function $\mathrm{Hint}(y) \in \{1, \ldots, K\}$ for some $K > 0$ such that for any $\varepsilon > 0$ and $N = \mathbf{H}_{\infty}(X, Y)$ the

event:

$$\forall_{y\in \mathrm{Hint}^{-1}(h)} \mathbf{H}_\infty\big(X|Y=y,\mathrm{Hint}(Y)=h\big)+\mathbf{H}_\infty\big(Y|\mathrm{Hint}(Y)=h\big) > \Big(1-\varepsilon-\frac{1}{K}\Big)\cdot N$$

occurs with probability $\geq 1-K\cdot 2^{-\varepsilon N}$ over the choice of h.

As a corollary, we significantly generalize this result in order to obtain the chain rule for many variables (see Corollary 2). To the best of our knowledge, prior to this work there was no efficient chain rule for min-entropy except Lemma 1 (see Dodis *et al.* [9] or Cachin and Maurer [6] for similar results for Renyi entropy).

It is important to mention that in any cryptographic application supplementary side information appears to be beneficial for the adversary and therefore using Lemma 3 should facilitate any security proof requiring detailed treatment of min-entropy (cf. Remark after proof of Lemma 5).

Usefulness of Chain rule. Our result seems to help to prove some facts that often look trivial at very first sight. The typical problem with proving such "obvious observations" comes from the fact that, in general, the chain rule for min-entropy is false. For example one may go through the proof of Lemma 5.2 from [5] to see how technical and delicate it is. We believe that using chain rule from this paper could significantly simplify reasonings like that. The reason for that is that our statement seems to work better then e.g. Lemma A.1 (min-entropy split) from [5] that is key for proving Lemma 5.2. The technical cause for that raises from the fact that Brakerski *et al.* split somehow the min-entropy of a pair but do not have full control of min-entropy of one of the elements. That makes the proof much harder in one case. Our techniques tend to follow the ideas from [5] and make them more clean to use.

Another good example for usefulness for our chain rule is Lemma 4 from this paper. Here we need the multivariate case of our result (Corollary 2) which was not considered in previous works at all.

3 Preliminaries

We assume the existence of a *random oracle*, i.e., perfectly random function $\mathcal{H}\colon \{0,1\}^* \to \{0,1\}^n$ which can be evaluated only by querying a certain oracle $\mathcal{H}$. At the beginning, all values of $\mathcal{H}$ are uniformly distributed, in particular, unpredictable. Throughout the protocols operation, one can issue a query $\mathcal{H}(m)$ obtaining the value of $\mathcal{H}(m)$ and gaining no other information.

In order to model leakage attacks, we introduce a *restricted leakage oracle* $\mathcal{O}^{\mathbb{D}}$ parametrized by a random variable $\mathbb{D} \in \{0,1\}^{|\mathbb{D}|}$. A query $\mathcal{O}^{\mathbb{D}}(f)$, consists of a function $f\colon \{0,1\}^{|\mathbb{D}|} \to \{0,1\}^\lambda$ given as a Turing machine, and results in the value $f(\mathbb{D})$. A *leakage oracle* $\mathcal{O}^{\mathbb{D},\mathcal{H}}$ (also denoted by $\mathcal{O}(\mathbb{D},\mathcal{H})$) is defined analogously but with leakage functions containing apart from ordinary operations a black-box access to a random oracle, which on an input x returns the value $\mathcal{H}(x)$. We say that the total leakage is of at most λ bits if the sum $\sum_i \lambda_i$ for all issued $f_i\colon \{0,1\}^{|\mathbb{D}|} \to \{0,1\}^{\lambda_i}$ is bounded by λ.

We denote by $\mathbf{TM}_\lambda^{\mathcal{O}(\mathbb{D})}$ the class of all probabilistic Turing machines equipped with an adaptive access to a restricted leakage oracle $\mathcal{O}$ with total leakage of at most

λ bits. Moreover, by $\mathbf{TM}_{\lambda,q}^{\mathcal{O}(\mathbb{M},\mathcal{H}),\mathcal{H}}$ we mean the subclass of $\mathbf{TM}_{\lambda}^{\mathcal{O}(\mathbb{M})}$ equipped with an adaptive access to a leakage oracle $\mathcal{O}^{\mathbb{D},\mathcal{H}}$ together with additional q executions of $\mathcal{H}$.

We say that a function f is $\mathcal{H}$-randomized (or simply randomized if no confusion can arise) if it is result is dependent on a certain random oracle $\mathcal{H}$. We denote a $\mathcal{H}$-randomized function by $f(-,\mathcal{H})$.

4 Chain rule for min-entropy

4.1 Bivariate case

We first prove the following case for two random variables.

Lemma 3. *Let X and Y be two (possibly dependent) random variables. Then, there exists a function $\mathrm{Hint}(y) \in \{1,\ldots,K\}$ for some $K > 0$ such that for any $\varepsilon > 0$ and $N = \mathbf{H}_\infty(X,Y)$ the event:*

$$\forall_{y\in \mathrm{Hint}^{-1}(h)} \mathbf{H}_\infty\big(X|Y=y, \mathrm{Hint}(Y)=h\big) + \mathbf{H}_\infty\big(Y|\mathrm{Hint}(Y)=h\big) > \Big(1-\varepsilon-\frac{1}{K}\Big)\cdot N$$

occurs with probability $\geq 1 - K\cdot 2^{-\varepsilon N}$ over the choice of h.

Proof. We begin with two straightforward facts concerning min-entropy. Firstly, observe that for any random variable Z and an event E the conditional min-entropy

$$\mathbf{H}_\infty(Z|E) \stackrel{\mathrm{def}}{=} \min_z\{-\log\left(\Pr(Z=z|E)\right)\}$$

satisfies the inequality $\mathbf{H}_\infty(Z|E) \geq \mathbf{H}_\infty(Z) - \log {}^1\!/_{\Pr(E)}$.

Secondly, using the formula for conditional probability we see that

$$\Pr(X=x|Y=y) = \frac{\Pr(X=x,Y=y)}{\Pr(Y=y)} \leq \frac{2^{-\mathbf{H}_\infty(X,Y)}}{\Pr(Y=y)},$$

which means that

$$\Pr(Y=y) \leq \frac{2^{-\mathbf{H}_\infty(X,Y)}}{\max_x \Pr(X=x|Y=y)} = 2^{-\mathbf{H}_\infty(X,Y)+\mathbf{H}_\infty(X|Y=y)}.$$

This consequently implies that $\mathbf{H}_\infty(Y) \geq \mathbf{H}_\infty(X,Y) - \max_y \mathbf{H}_\infty(X \mid Y=y)$.

We now proceed to the proof of Lemma 3. We define a function Hint by the condition:

$$\mathrm{Hint}(y) = i \iff \mathbf{H}_\infty(X|Y=y) \in [\frac{i-1}{K}N, \frac{i}{K}N],$$

where N denotes the min-entropy $\mathbf{H}_\infty(X,Y)$ (we disregard the boundary cases). By the definition of Hint we see that

$$\forall_{y\in \mathrm{Hint}^{-1}(i)} \mathbf{H}_\infty(X|\mathrm{Hint}=i, Y=y) \geq \frac{i-1}{K}N. \tag{2}$$

Moreover, using both of the above general observations, we get that

$$\mathbf{H}_\infty(Y|\text{Hint}=i) \geq \mathbf{H}_\infty(X,Y|\text{Hint}=i) - \frac{i}{K}N \geq N - \frac{i}{K}N - \log {}^1\!/_{\Pr(\text{Hint}=i)}.$$

By summing (2) and (4.1) up we obtain:

$$\forall_{y\in\text{Hint}^{-1}(i)}\mathbf{H}_\infty(X|\text{Hint}=i,Y=y) + \mathbf{H}_\infty(Y|\text{Hint}=i) \geq N - \frac{N}{K} - \log {}^1\!/_{\Pr(\text{Hint}=i)}.$$

Now observe that for all values i of Hint that satisfy $\Pr(\text{Hint}=i) \geq 2^{-\varepsilon N}$ we have:

$$\forall_{y\in\text{Hint}^{-1}(i)}\mathbf{H}_\infty(X|\text{Hint}=i,Y=y)+\mathbf{H}_\infty(Y|\text{Hint}=i) \geq N-\frac{N}{K}-\varepsilon N = (1-\frac{1}{K}-\varepsilon)\cdot N.$$

There are at most K other values of Hint which consequently occurs with probability smaller than $K \cdot 2^{-\varepsilon N}$. This finishes the proof. □

Remark 2. For the sake of brevity, from now on we omit the $\forall$ quantifier and write $\mathbf{H}_\infty(X; Y=y, \text{Hint}=h)$ to denote the fact that y is consistent with the value h of a random variable $\text{Hint}=\text{Hint}(Y)$. Then, with the same assumptions as above, we have [1]:

$$\widetilde{\mathbf{H}}_\infty(X|Y;\text{Hint}=h) + \mathbf{H}_\infty(Y|\text{Hint}=h) > \left(1-\varepsilon-\frac{1}{K}\right)\cdot\mathbf{H}_\infty(X,Y) \qquad (3)$$

$$\mathbf{H}_\infty(X|\text{Hint}=h,E) + \mathbf{H}_\infty(Y|\text{Hint}=h) > \left(1-\varepsilon-\frac{1}{K}\right)\cdot\mathbf{H}_\infty(X,Y) \qquad (4)$$

for any event E depending only on Y with probability $\geq 1 - K \cdot 2^{-\varepsilon N}$ over the choice of h. This follows from averaging over the choice of y. Note that the subtlety of definition of conditional min-entropy given in [9] is not relevant as the formula (4.1) works for any y such that $\text{Hint}(y) = h$.

As corollaries we obtain:

Corollary 1. *Let X, Y be random variables satisfying $\mathbf{H}_\infty(X,Y) = mn$. For every $\Delta \in \mathbb{N}$ such that there exists a random variable* Hint *such that:*

$$\mathbf{H}_\infty(X|Y=y,\text{Hint}=h) + \mathbf{H}_\infty(Y|\text{Hint}=h) > (m-2\Delta)n$$

occurs with probability $\geq 1 - 2^{-(\Delta n - \log m)}$ over the choice of h.

Proof. Apply Lemma 3 for $\varepsilon = \frac{\Delta}{m}$ and $K = \lceil \frac{m}{\Delta} \rceil$.

4.2 Multivariate case

Moreover, by more involved inductive considerations we obtain:

[1] By $\widetilde{\mathbf{H}}_\infty(X|Y;\text{Hint}=h)$ we mean the conditional min-entropy $\widetilde{\mathbf{H}}_\infty(X|Y)$ computed with respect to the distribution conditioned on the event $\text{Hint}=h$

Corollary 2. *Let $X_1, \ldots, X_\ell$ be random variables satisfying $\mathbf{H}_\infty(X_1, \ldots, X_\ell) = s\ell$ for some $s > 0$. Then, for any $D > 0$ there exists a random variable* Hint *such that*

$$\sum_{1 \le i \le \ell} \mathbf{H}_\infty(X_i | \mathrm{Hint} = h, E_i) \ge s\ell(1 - \frac{1}{D}) \tag{5}$$

with probability $1 - 2D\ell(\ell-1) \cdot 2^{-\frac{s}{2D}}$ over the choice of h, where E_i are events depending on Hint *and variables with smaller indices, i.e., $X_1, \ldots, X_i$.*

Proof. We prove the following claim by descending induction with respect to k.

Claim. For any $k \in [1, \ell]$ there exists a random variable Hint_k such that:

$$\mathbf{H}_\infty(X_1, \ldots, X_k | \mathrm{Hint}_k = h) + \sum_{k+1 \le i \le \ell} \mathbf{H}_\infty(X_i | \mathrm{Hint} = h) \ge s(\ell - \frac{\ell - k}{D}) \tag{6}$$

with probability $1 - 2D\ell(\ell-k) \cdot 2^{-\frac{s}{2D}}$ even if we additionally condition $\mathbf{H}_\infty(X_{k+m}; \mathrm{Hint}_k = i)$ for $m > 0$ with any event E_{k+m} depending solely on Hint_k and variables with smaller indices, i.e., $X_1, \ldots, X_{k+m-1}$.

Proof (Proof of the claim). The base case $k = \ell$ is just the assumption $\mathbf{H}_\infty(X_1, \ldots, X_\ell) = s\ell$ for an empty variable Hint_ℓ (note that there is no additional assumption on further conditioning). Now, assume that the claim is true for some $k > 1$ and that

$$\mathbf{H}_\infty(X_1, \ldots, X_k | \mathrm{Hint}_k = i) = c_i s,$$

for some $0 < c_i \le \ell$. By Lemma 3 applied for $\varepsilon = \frac{1}{2Dc_i}$, $K = 2D\lceil c_i \rceil$, $X = X_k$, and $Y = (X_1, \ldots, X_{k-1})$ conditioned on $\mathrm{Hint}_k = i$ we obtain a random variable H_i^k such that

$$\begin{aligned} \mathbf{H}_\infty(X_k | \mathrm{Hint}_k = i, \mathrm{H}_i^k = h', E) + \mathbf{H}_\infty(Y | \mathrm{Hint}_k = i, \mathrm{H}_i^k = h') &\ge c_i s(1 - \frac{1}{2Dc_i} - \frac{1}{2D\lceil c_i \rceil}) \\ &\ge s(c_i - \frac{1}{D}) = sc_i - \frac{s}{D}. \end{aligned} \tag{7}$$

for any event E depending on Y with probability $1-2D\lceil c_i\rceil\cdot 2^{-\frac{s}{2D}} \geq 1-2D\ell\cdot 2^{-\frac{s}{2D}}$ over the choice of h'. We now set $\mathrm{Hint}_{k-1} = (\mathrm{Hint}_k, \mathrm{H}^k_{\mathrm{Hint}_k})$ and compute that

$$\mathbf{H}_\infty(X_1,\ldots,X_{k-1}|\mathrm{Hint}_{k-1}=(h,g)) + \sum_{k\leq i\leq \ell} \mathbf{H}_\infty(X_i|\mathrm{Hint}_{k-1}=(h,g))) \geq$$

$$\mathbf{H}_\infty(Y;\mathrm{Hint}_{k-1}=(h,g)) + \mathbf{H}_\infty(X_k|\mathrm{Hint}_{k-1}=(h,g)) + \sum_{k+1\leq i\leq \ell} \mathbf{H}_\infty(X_i|\mathrm{Hint}_{k-1}=(h,g)) \geq \tag{8}$$

$$\mathbf{H}_\infty(X_1,\ldots,X_k|\mathrm{Hint}_k=h) - \frac{s}{D} + \sum_{k+1\leq i\leq \ell} \mathbf{H}_\infty(X_i|\mathrm{Hint}_{k-1}=(h,g)) \geq \tag{9}$$

$$\mathbf{H}_\infty(X_1,\ldots,X_k|\mathrm{Hint}_k=h) - \frac{s}{D} + \sum_{k+1\leq i\leq \ell} \mathbf{H}_\infty(X_i|\mathrm{Hint}_k=h) \geq \tag{10}$$

$$\geq s(\ell - \frac{\ell-k+1}{D})$$

with probability

$$\geq (1-2D\ell(\ell-k)\cdot 2^{-\frac{s}{2D}})(1-2D\ell\cdot 2^{-\frac{s}{2D}}) \geq 1-2\ell(\ell-k+1)\cdot 2^{-\frac{s}{2D}},$$

where in (8) we used the definition of Y, in (9) we applied the formula (7) and in (10) we removed additional conditioning on event $\mathrm{H}^k_{\mathrm{Hint}_k} = g$ which depends only on variables with smaller indices (induction step) and is therefore harmless (cf. hypothesis of the claim). The inductive claim concerning additional conditioning follows along the same lines (just by adding further events in min-entropies), but is a bit cumbersome to state succinctly.

Our corollary follows from the claim for $k=1$ and the inequality $\frac{\ell-1}{\ell} < 1$, i.e., the expected Hint equals Hint_1.

Remark 3. The proof above shows that the size of Hint is in fact polynomial in ℓ and D.

We now state an elementary proposition which allows us to obtain a certain bound on the number of high min-entropy blocks given $\mathbf{H}_\infty(X_1,\ldots,X_\ell) \geq \beta\ell n$ for $\beta \leq 1$.

Proposition 1. *Let $x_1,\ldots,x_\ell$ be a sequence of numbers satisfying $0 \leq x_i \leq n$ and $x_1+\cdots+x_\ell = \beta\ell n$ for some $0 < \beta \leq 1$. Then, for any $0 \leq \gamma < \beta$ there are more than $\lfloor \frac{\beta-\gamma}{1-\gamma}\ell \rfloor$ numbers x_i such that $x_i \geq \gamma n$.*

A proof of this fact appears in Appendix C. As a corollary we obtain:

Corollary 3 (High min-entropy blocks). *Let $X_1,\ldots,X_\ell$ be random variables distributed over $\{0,1\}^n$ and satisfying $\mathbf{H}_\infty(X_1,\ldots,X_\ell) \geq \beta\ell n$. Then, for any D and $\gamma < \beta(1-\frac{1}{D})$ there exists a random variable* Hint *such that with probability $1-2D\ell(\ell-1)\cdot 2^{-\frac{\beta n}{2D}}$ the number of blocks X_i satisfying $\mathbf{H}_\infty(X_i|\mathrm{Hint}=h) \geq \gamma n$ is greater than $\lfloor \frac{\beta(1-\frac{1}{D})-\gamma}{1-\gamma}\ell \rfloor$. In particular, for $D=2$ and $\gamma = \frac{\beta}{4}$ we obtain that there*

exists a random variable Hint *such that with probability* $1 - 4\ell(\ell-1)\cdot 2^{-\frac{\beta n}{4}}$ *there exists* $\lfloor \frac{\beta}{4-\beta}\ell \rfloor \geq \lfloor \frac{\beta\ell}{4} \rfloor$ *blocks of min-entropy* $\mathbf{H}_\infty(X_i|\text{Hint} = h) \geq \frac{\beta n}{4}$.

Proof. For the general case, we consequently use Corollary 2 and Proposition 1. We obtain the special case by direct specialization.

5 Key derivation procedure based on sensitive data

In this chapter, we define formally a class of protocols whose security can be expressed in terms of a game with a certain probability of success. Consequently, we define two properties, security and privacy, of a randomised transformation function $\mathsf{kdp}(-, \mathcal{H})$ which makes it suitable for derivation of keys from sensitive disk data.

5.1 Security games

Definition 1 (Security game). *Let K be a random variable. A* security game against an adversary $\mathcal{A}$ based on randomness K *is a tuple* $\mathsf{Game} = (\mathcal{C}, \mathsf{KeyGen}, \mathsf{Setup}_{\text{Ch}}, \mathsf{Setup}_{\text{Adv}}, \mathsf{Execute})$ *consisting of an interactive algorithm $\mathcal{C}$ together with a randomized key generation procedure* KeyGen*, a pair of setup procedures* $\mathsf{Setup}_{\text{Ch}}$, $\mathsf{Setup}_{\text{Adv}}$ *and an execution procedure* Execute *which given an interactive algorithm $\mathcal{A}$ operates as described in Fig. 1 below.*

Execution procedure Execute **of** $\mathsf{Game} = (\mathcal{C}, \mathsf{KeyGen}, \mathsf{Setup}_{\text{Ch}}, \mathsf{Setup}_{\text{Adv}})$.

1. The key is initialized by $\texttt{key} \leftarrow \mathsf{KeyGen}(K)$ and then the input tapes of $\mathcal{A}$ and $\mathcal{C}$ are set to $\mathsf{Setup}_{\text{Adv}}(\texttt{key})$ and $\mathsf{Setup}_{\text{Ch}}(\texttt{key})$ respectively.

Execution phase:

2. The following loop is conducted:

Algorithm 1: Main loop

```
1 msg_A = ⊥ /* first message is empty                    */
2 while state_C ∉ {Accept, Reject} do
3 |   A : (state_A, msg_A) ⇝ (state'_A, msg_C)
4 |   C : (state_C, msg_C) ⇝ (state'_C, msg_A)
5 end
```

3. When $\mathcal{C}$ transits to either `Accept` or `Reject` the execution terminates.

Output: The final state $\mathsf{state}_{\mathcal{C}} \in \{\texttt{Accept}, \texttt{Reject}\}$ of the challenger $\mathcal{C}$.

Fig. 1. Execution procedure

In lines 3 and 4 of the Main Loop we used the notation $\mathcal{B} : (\texttt{state}, \texttt{msg}) \rightsquigarrow (\texttt{state}', \texttt{msg}')$ *which indicates that the interactive machines $\mathcal{B}$ resumes in state* `state` *given an input message* `msg` *and transits to state* $\texttt{state}'$ *with an output message* $\texttt{msg}'$*. Given all the parameters, we denote the execution of the game by* $\mathsf{Game}[\mathcal{A} \leftrightharpoons \mathcal{C}, \texttt{key} \leftarrow \mathsf{KeyGen}(K)]$.

Intuitively, the operation of $\mathsf{Game}[\mathcal{A} \leftrightharpoons \mathcal{C}, \mathtt{key} \leftarrow \mathsf{KeyGen}(K)]$ boils down to an adaptive, sequential (numbered by round) exchange of messages between interactive machines $\mathcal{A}$ and $\mathcal{C}$ initialized by the values $\mathsf{Setup}_{\mathrm{Adv}}(\mathtt{key})$ and $\mathsf{Setup}_{\mathrm{Ch}}(\mathtt{key})$ respectively, which ends up in the last state of the algorithm $\mathcal{C}$.

Remark 4 (Complexity). We can require that $\mathcal{C}$ or $\mathcal{A}$ belong to a certain complexity class **TM** which might be characterized by number of Turing machine steps allowed to be taken, access to some oracle $\mathcal{O}$ (e.g., hash, leakage) or a memory bound. For example, to describe an adversary which works in time polynomial in the size of the key $\mathtt{key}$ and can access leakage oracle $\mathcal{O}(\mathtt{key})$ with a leakage bound λ, we write that $\mathcal{A} \in \mathbf{TM}_{\lambda}^{\mathcal{O}(\mathtt{key})}(\mathrm{poly}(|\mathtt{key}|))$. While computing any kind of complexity (e.g. time, storage, leakage), as a final result we consider the total amount of resources used during all transitions conducted (cf., e.g., line 3 and 4 above) in an interactive algorithm. In particular, an adversary $\mathcal{A}$ belongs to $\mathbf{TM}_{\lambda}^{\mathcal{O}(\mathtt{key})}$ if the total amount of leakage obtained during *all transitions* of $\mathcal{A}$ does not exceed λ bits.

Remark 5 (Relation to classical Interactive Turing Machines). In fact the definition given above is a RAM-based analogue of interactive Turing machines. We resigned from the formal approach based on common input tapes assumption in order to emphasize the sequential nature of the computation, which we shall exploit in the upcoming considerations.

The above security game is tailored to cover a broad range of security definitions of various cryptographic protocols. We now state the description of a class of cryptographic protocols whose security is grasped through game-based definition.

Definition 2 (Game security). *We say that* $\mathsf{Game} = (\mathcal{C}, \mathsf{KeyGen}, \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$ *based on randomness* K *is* $(\varepsilon, \mathbf{TM})$*-secure iff for every* $\mathcal{A} \in \mathbf{TM}$ *the probability that execution ends in* $\mathtt{Accept}$ *satisfies*

$$\Pr(\mathsf{Game}[\mathcal{A} \leftrightharpoons \mathcal{C}, \mathtt{key} \leftarrow \mathsf{KeyGen}(K)] = \mathtt{Accept}) \leqslant \varepsilon,$$

where the probability is taken over K *and all random choices of* $\mathcal{C}$ *and* $\mathcal{A}$*.*

Example 3. The vast majority of cryptographic protocols are covered by the above game-based definition. Good example is identification scheme in BRM from [1]. Another protocol that fits the definition is described in the introduction standard Merkle-tree authentication protocol.

Note, that the notation ε for the security parameter might be confusing as we do not distinguish the case of unpredictability and indistinguishability applications (see [10] for precise definitions), i.e., the security definition above covers the case of $\varepsilon \approx 0$ and $\varepsilon \approx \frac{1}{2}$.

5.2 Security and privacy of key derivation functions

After describing what security games are, we are ready to formulate precise definitions for intuitive requirements of privacy and security that we impose on our key-derivation procedure. From now on, D is a random variable representing disk data, λ denotes the number of bits adversary can leak, N is the maximal value of min-entropy of disk data and p is the ratio of actual min-entropy $\mathbf{H}_{\infty}(D)$ and N.

Definition 3 (privacy of a key-derivation procedure). *We say that a randomized function* $\mathsf{kdp}(-,\mathcal{H}) : \{0,1\}^N \to \{0,1\}^M$ *is* $(p,\lambda,\Delta_\lambda,q,\varepsilon)$*-private if there exists a simulator* $\mathcal{S} \in \mathbf{TM}^{\mathcal{O}(D)}_{\lambda+\Delta_\lambda}$ *such that for every random variable* $D \in \{0,1\}^N$ *of min-entropy* $\mathbf{H}_\infty(D) \geq pN$ *and every adversary* $\mathcal{A} \in \mathbf{TM}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}_{\lambda,q}$ *operating on* $\mathtt{key} = \mathsf{kdp}(D,\mathcal{H})$*, the output distributions satisfy:*

$$(\mathbf{Output}(\mathcal{A}(\mathtt{key})), D) \approx_\varepsilon (\mathbf{Output}(\mathcal{S}(\mathcal{A})), D).$$

The privacy definition tracks down the amount of additional leakage Δ_λ that is necessary to constructor $\mathcal{S}$ capable of simulating the behaviour of any adversary $\mathcal{A}^{\mathcal{O},\mathcal{H}}_{\lambda,q}$ operating on the key generated by the dispersing procedure. Observe that any algorithm $\mathcal{A}(\mathtt{key}) \in \mathbf{TM}^{\mathcal{O}(D),\mathcal{H}}_{\lambda,q}$ (cf. Section 3 for the formal specification of this complexity class) is provided with access to an oracle $\mathcal{H}$, i.e., can test values of a random function, and moreover issues a sequence of leakage queries $\mathcal{O}^{D,\mathcal{H}}(f_i)$, which may also depend on the random oracle $\mathcal{H}$, i.e., can learn some information concerning D depending on the same random function $\mathcal{H}$.

Definition 4 (Security of a key-derivation procedure). *We say that a randomized function* $\mathsf{kdp}(-,\mathcal{H}) : \{0,1\}^N \to \{0,1\}^M$ *is* $(p,\lambda,\Delta_\lambda,q,\varepsilon)$*-secure if for any* $(\varepsilon', \mathbf{TM}^{\mathcal{O}(\mathtt{key})}_{\lambda})$*-secure game* $\mathsf{Game} = (\mathcal{C}, \mathsf{id}, \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$[2] *for randomness* $K \leftarrow U_N$*, the game* $\mathsf{Game}_{\mathrm{Disk}} = (\mathcal{C}, \mathsf{kdp}(D,\mathcal{H}), \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$ *based on randomness* $(D,\mathcal{H})$ *is* $(\varepsilon'+\varepsilon, \mathbf{TM}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}_{\lambda-\Delta_\lambda,q})$*-secure.*

The general aim of the security definition is to grasp the intuitive expectation that the an adversary playing against key derived from sensitive data should not gain any advantage comparing to the case of using a redundant, truly random key. Note that it suffices to give $\mathcal{A}$ the access to $\mathtt{key} = \mathsf{kdp}(D,\mathcal{H})$ as the transcript of any scheme's execution can be generated based on $\mathtt{key}$.

6 Disperse as a key derivation procedure

6.1 Disperse graph

Throughout the whole construction we shall make use of bipartite right M-regular graphs identified with functions $\sigma : [N_1] \times [M] \to [N_0]$ by the following recipe. By $\mathcal{G}_\sigma$ we denote a bipartite graph $\mathcal{G}$ with the sets of vertices equal to two disjoint sets $[N_0], [N_1]$ and with edges going from $n \in [N_1]$ to $\sigma^n_m \in [N_0]$ for any $m \in [M]$. The following definition is crucial:

Definition 5. *A bipartite graph* $\mathcal{G} = (V^0 \sqcup V^1, E)$ *is a* right (K,L)-disperser *if for every set* $S \subset V^1$ *such that* $|S| = K$ *the neighbourhood* $N(S)$ *satisfies*

$$|N(S)| \geqslant L,$$

i.e. the sets of size K *expands into sets of size at least* L*.*

[2] for detailed description of Game see Definition 1

We often make use of explicit ℓ^d-regular $(\ell^e, (1-\eta)\ell)$-dispersers. We implicitly assume that the numbers d, e satisfy $d < 1, e < 1$ and $d + e > 1$. For more details on dispersers and further definitions see Appendix B.

In the Fig. 2, we describe function Disperse explicitly. For the sake of simplicity, we identify vertices of graph with labels they contain. An exemplary Disperse function is shown in Fig. 3.

Implementation of $\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H})$.

Input: a bitstring $D = D_1 \dots D_\ell$ for $D_1, \dots, D_\ell \in \{0,1\}^n$; $\mathcal{G}_\sigma$ a d-regular biparite graph $(D \sqcup D', E)$, where $D = (D_1, \dots, D_\ell)$ and $D' = (D'_1, \dots, D'_\ell)$, such that $N(D'_i) = \{D_{\sigma_1^i}, \dots, D_{\sigma_d^i}\}$; function $\mathcal{H}\colon \{0,1\}^{dn+\log \ell} \to \{0,1\}^n$
Output: a bitstring D'.

Execution:

1. Assign values to the "upper" vertices of $\mathcal{G}_\sigma$:
   $D'_i \leftarrow \mathcal{H}\left(i, D_{\sigma_1^i}, \dots, D_{\sigma_d^i}\right)$, for $i = 1, \dots, \ell$.
2. Return $D' = D'_1, \dots, D'_\ell$.

Fig. 2. Operation of dispersion function.

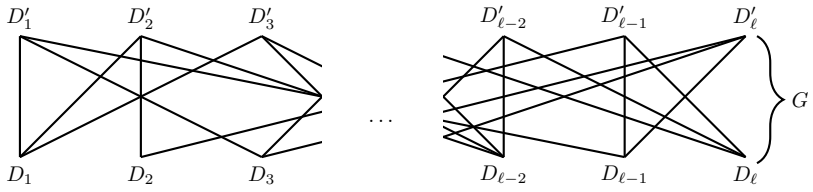

Fig. 3. An exemplary $\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H})$

We now state our main result that for an appropriately chosen graph $\mathcal{G}_\sigma$ the function $\mathsf{Disperse}_{\mathcal{G}_\sigma}$ is in fact private and secure for reasonable parameters.

Theorem 1. *Let $\mathcal{G}_\sigma$ be a ℓ^d-regular $(\ell^e, (1-\eta)\ell)$-right disperser and $\mathcal{H}$ a random oracle. Then for any β satisfying $p - \frac{\lambda}{\ell n} > \beta > 4\eta$ the function $\mathsf{Disperse}_{\mathcal{G}_\sigma}(-, \mathcal{H}) : \{0,1\}^{\ell n} \to \{0,1\}^{\ell n}$ is:*

- *$(p, \lambda, \Delta_\lambda = \ell^e(\log q + \log \ell), q = 2^{o(\ell^{1-e})n}, O(\ell^2 \cdot 2^{\frac{-\beta n}{4}}))$-private*
- *$(p, \lambda, \Delta_\lambda = \ell^e(\log q + n), q = 2^{o(\ell^{1-e})n}, \varepsilon = O(\ell^2 \cdot 2^{\frac{-\beta n}{4}}))$-secure.*

Since the proof of this theorem is long, it is divided into three parts – at the beginning it is shown that even under the presence of leakage, function Disperse effectively hides the data underneath; then, basing on this result, privacy property is proven; next security is shown. At the end, we also shortly elaborate about the efficiency of $\mathsf{Disperse}_{\mathcal{G}_\sigma}$.

Before proceeding to actual proofs we shortly elaborate about the bounds on the parameters.

Remark 6 (Efficiency of $\mathsf{Disperse}_{\mathcal{G}_\sigma}$*).* It is important to note that in order to obtain a single bit of a derived key one need process ℓ^d blocks of disk data. This therefore constitutes a leakage-time trade-off for the operation of our function. Namely reduction of d allows to compute a single bit of key more efficiently with a cost of an increased parameter Δ_λ proportional to ℓ^e (recall that $d + e > 1$).

Remark 7 (Bounds on parameters). The bounds $p - \frac{\lambda}{\ell n} > \beta$, resp. $\beta > 4\eta$ express natural requirements that leakage ratio $\frac{\lambda}{\ell n}$ should not exceed the actual ratio p, resp. the quality of disperser η should be superior to the entropy reserve represented by $p - \frac{\lambda}{\ell n}$. The bound on $q = 2^{o(\ell^{1-e})n}$ corresponds to a robust, exponential bound on the random oracle query-based complexity of an adversary.

6.2 One-wayness of Disperse

The $\mathsf{Disperse}$ procedure possesses a certain *one-wayness* property expressed in the following lemma. We precede it with a necessary definition.

Definition 6 (Bad query). *Given a random variable D, a bipartite right d-regular graph $\mathcal{G}_\sigma$ and a random oracle $\mathcal{H}$ we say that a random oracle query $\mathcal{H}(b)$, submitted by some Turing Machine $\mathcal{A}$, is* bad *if the argument b equals $(i, D_{\sigma_1^i} D_{\sigma_2^i} \dots D_{\sigma_d^i})$ for some $i \in \{1, \dots, \ell\}$, i.e., the argument of random oracle query equals one of the values defined by graph $\mathcal{G}_\sigma$ and a random variable D.*

By $\mathrm{Bad}_{\mathcal{A}}$ we denote the set of all bad queries. By $\mathsf{indices}_{\mathcal{A}}$ we denote a list of all pairs (k, i_k) of indices $k \in \{1, \dots, q\}$ and $i_k \in \{1, \dots, \ell\}$ such that k is the smallest index of a bad random oracle query of $\mathcal{A}$ which is equal to $(i_k, D_{\sigma_1^{i_k}} \dots D_{\sigma_\delta^{i_k}})$. Since the total number of queries is q and $\mathcal{G}_\sigma$ has 2ℓ vertices, we can describe the list $\mathsf{indices}_{\mathcal{A}}$ using $|\mathsf{indices}_{\mathcal{A}}| \cdot (\log \ell + \log q)$ bits.

Lemma 4 (One-wayness of $\mathsf{Disperse}$**).** *Let $\mathcal{G}_\sigma$ be a ℓ^d-regular $(\ell^e, (1-\eta)\ell)$-right disperser and $D = (D_1, \dots, D_\ell) \in \{0,1\}^{n\ell}$ be a random variable of min-entropy $p\ell n$. Then, the probability that an algorithm $\mathcal{A}(\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H})) \in \mathbf{TM}_{\lambda,q}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}$ makes at least ℓ^e different bad queries satisfies:*

$$\Pr(|\mathsf{indices}_{\mathcal{A}}| \geq \ell^e) = O(\ell^2 \cdot 2^{\frac{-\beta n}{4}})$$

for any β satisfying $p - \frac{\lambda}{\ell n} > \beta > 4\eta$ and $q = 2^{o(\ell^{1-e})n}$.

Proof. Due to technical nature the proof is deferred to Appendix A.2.

6.3 Privacy of Disperse

In this section we show that $\mathsf{Disperse}$ is in fact a private key-derivation procedure. The bottom line of the proof is an application of one-wayness together with a careful design of leakage query. It is important to note that we significantly use our computational model, where we can submit potentially non-polynomial queries.

Theorem 2 (Privacy). *Let $\mathcal{G}_\sigma$ be a ℓ^d-regular $(\ell^e, (1-\eta)\ell)$-right disperser and $D = (D_1, \dots, D_\ell) \in \{0,1\}^{n\ell}$ be a random variable of min-entropy $p\ell n$. Then,*

there exists a simulator $\mathcal{S} \in \mathbf{TM}^{\mathcal{O}(D)}_{\lambda+\ell^e(\log q+\log \ell)}$ *such that for every adversary* $\mathcal{A} \in \mathbf{TM}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}_{\lambda,q}$ *operating on the key* $\mathsf{key} = \mathsf{Disperse}_{\mathcal{G}_\sigma}(D,\mathcal{H})$*, the output distributions satisfy:*

$$(\mathbf{Output}(\mathcal{A}(\mathtt{key})), D) \approx_\varepsilon (\mathbf{Output}(\mathcal{S}(\mathcal{A})), D)$$

for $\varepsilon = O(\ell^2 \cdot 2^{\frac{-\beta n}{4}})$*,* $q = 2^{o(\ell^{1-e})n}$ *and any* β *satisfying* $p - \frac{\lambda}{\ell n} > \beta > 4\eta$*.*

In order to give a proof, we shall construct a machine $\mathcal{S}$ such that for any adversary $\mathcal{A}(\mathsf{Disperse}_{\mathcal{G}_\sigma}(D,\mathcal{H})) \in \mathbf{TM}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}_{\lambda,q}$ the result of $\mathcal{S}(\mathcal{A})$ is indistinguishable from $\mathcal{A}(K)$ conditioned on D. We precede the construction by an essential transformation of random oracles and leakage functions, which plays a role of random oracle re-programming.

Definition 7 (Twisted random oracle). *Let* $\mathcal{H}$ *be a random oracle and* $L = \langle(\arg_1, v_1), \ldots, (\arg_k, v_k)\rangle$ *be a list of pairs of an argument* $\arg_i$ *together with a potential value* v_i*. We define a* twisted random oracle $\mathcal{H}\{L\}$ *to be an oracle whose operation is described as follows:*

$$\mathcal{H}\{L\}(q) = \begin{cases} v_i & \text{if } q = \arg_i \text{ for some } i \\ \mathcal{H}(q) & \text{otherwise.} \end{cases}$$

In particular, given a random variable D*, a random oracle* $\mathcal{H}$ *and a random variable* $K = \langle K_1, \ldots, K_\ell\rangle \in \{0,1\}^{\ell n}$*, by* $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}$ *we denote a random oracle* $\mathcal{H}\{\langle(D_{\sigma_i^1} \ldots D_{\sigma_i^{\deg(G)}}, K_i)\rangle_{i=1\ldots\ell}\}$*. Observe that if* $K \sim U_{\ell n}$ *is independent of* $\mathcal{H}$ *then the distributions of* $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}$ *and* $\mathcal{H}$ *are the same.*

Construction of the simulator The operation of $\mathcal{S}(\mathcal{A})$, based on the description of $\mathcal{A}$, consists of the following steps described in Fig. 4.

Before giving a formal proof of statistical indistinguishability of output distributions, we give some claryifing remarks about consecutive steps of the construction. Firstly, we should emphasize that in Step (2) we crucially use the properties of our leakage model by querying leakage oracle with potentially non-polynomial function simulating whole behaviour of $\mathcal{A}$. Secondly, observe that in Step (2) the simulator leaks only the indices of queries, not their actual arguments as those can be observed during Step (3) of simulation. Thirdly, note that in Step (3a) we need not perform any additional leakage apart from the value of f, as $f\{D \xrightarrow{\mathcal{G}_\sigma} K\}$ can be obtained inside the leakage query as in Step (2). Therefore the leakage excess consists merely of the list $\mathsf{indices}_\mathcal{A}$ and consequently $\Delta\lambda = |\mathsf{indices}_\mathcal{A}|(\log q + \log \ell)$.

Proof (Proof of Theorem 2). We shall now argue that the simulator $\mathcal{S}$ constructed above satisfies the requirements of Theorem 2 for any adversary $\mathcal{A}$. Concretely, we prove that $\mathcal{S}$ perfectly simulates the execution of any adversary $\mathcal{A}$, unless $|\mathsf{indices}_\mathcal{A}| \geq \ell^e$. Therefore, for any adversary $\mathcal{A}$ the output's distribution of $\mathcal{S}(\mathcal{A})$ satisfies:

$$(\mathbf{Output}(\mathcal{A}(K)), D) \approx_\varepsilon (\mathbf{Output}(\mathcal{S}(\mathcal{A})), D),$$

Implementation of the simulator $\mathcal{S}$.

1. $\mathcal{S}$ initializes a random oracle $\mathcal{H}$, i.e., creates a table $\mathcal{H}$ of uniformly random values associated to all inputs of $\mathcal{H}$ (or use **OracleQueryList**). Moreover, it draws a random variable $K \leftarrow U_{\ell n}$.
2. $\mathcal{S}$ initializes the random tape of $\mathcal{A}$ to a fixed sequence of uniformly random bits and then queries the leakage oracle with the Turing machine $\mathsf{ind} : \{0,1\}^{|D|} \to \{0,1\}^*$ which operates as follows:

   Operation of ind:

   Description of the function Simulate the execution of $\mathcal{A}(K)$ step by step with random oracle queries $\mathcal{H}$ substituted with $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}$. Every time the adversary $\mathcal{A}$ issues a leakage oracle query given by a Turing machine f, the simulator $\mathcal{S}$ provides her with a result of a *twisted leakage function* $f\{D \xrightarrow{\mathcal{G}_\sigma} K\}$, i.e., a Turing machine with all random oracle queries substituted with $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}$.
   Result The list $\mathsf{indices}_\mathcal{A}$. Returns $\mathsf{indices}_\mathcal{A}$ if its length satisfies $|\mathsf{indices}_\mathcal{A}| < \ell^e$, or $\perp$ otherwise.
   Complexity Leakage: $|\mathsf{indices}_\mathcal{A}|(\log q + \log \ell)$
3. $\mathcal{S}$ executes $\mathcal{A}(K)$ with a previously initialized (see Step (2)) random tape and $\mathcal{H}$ sampled above (see Step (1)), and then runs it step by step with the following exceptions:
   (a) When $\mathcal{A}$ issues a leakage query given by a Turing machine f, the simulator $\mathcal{S}$ substitutes it with a twisted leakage function $f\{D \xrightarrow{\mathcal{G}_\sigma} K\}$.
   (b) $\mathcal{S}$ keeps track of the number k of random oracle queries issued to $\mathcal{H}$ and every time it appears in a pair $(k, i_k) \in \mathsf{indices}_\mathcal{A}$, replaces the value returned by $\mathcal{H}$ with K_{i_k}. Moreover, it stores the arguments a_k of queries appearing in the list $\mathsf{indices}_\mathcal{A}$ and substitutes the value of $\mathcal{H}$ with K_{i_k} every time a_k appears as an argument.

Fig. 4. Implementation of Simulator

where $\varepsilon = \Pr(|\mathsf{indices}_\mathcal{A}| \geq \ell^e)$. Firstly, note that the execution of $\mathcal{A}$ inside the leakage function ind_u (see Step (2)) is perfectly equivalent to an honest execution of $\mathcal{A}$ as $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}$ is distributed equally to $\mathcal{H}$. Consequently, the actual simulation given in Step (3) differs from a perfect simulation only by the condition on $|\mathsf{indices}_\mathcal{A}|$, as its perfectly equivalent to the one performed during simulators leakage phase. This condition forces the return of $\perp$ instead of appropriate $\mathsf{indices}_\mathcal{A}$ with probability $\Pr(|\mathsf{indices}_\mathcal{A}| \geq \ell^e) = \varepsilon$. Consequently, we bound ε by a factor negligible (in a certain sense) in the security parameters. Directly by applying Lemma 4 for an adversary $\mathcal{A}$ we see that:

$$\varepsilon = \Pr(|\mathsf{indices}_\mathcal{A}| \geq \ell^e) = O(\ell^2 \cdot 2^{\frac{-\beta n}{4}}).$$

This completes the proof.

6.4 Security of Disperse

Again, based on one-wanness of Disperse we prove that our function satisfies the security requirements. We have the following:

Theorem 3 (Security). *Let* $\mathsf{Game} = (\mathcal{C}, \mathsf{id}, \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$[3] *be an* ε*-secure game based on randomness* $K \sim U_{n\ell}$ *for the class of adversaries* $\mathbf{TM}_{\lambda}^{\mathcal{O}(K)}$ *then for every* ℓ^d*-regular* $(\ell^e, (1-\eta)\ell)$*-disperser* $\mathcal{G}_\sigma$*,* β *satisfying* $p - \frac{\lambda}{\ell n} > \beta > 4\eta$ *and* $q = 2^{o(\ell^{1-e})n}$*, the game* $\mathsf{Game}_{\mathrm{Disk}} = (\mathcal{C}, \mathsf{Disperse}_{\mathcal{G}_\sigma}, \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$ *based on randomness* $(D, \mathcal{H})$ *of min-entropy* $\mathbf{H}_\infty(D) = p\ell n$ *is* $\varepsilon + O(\ell^2 \cdot 2^{\frac{-\beta n}{4}})$*-secure for adversaries in* $\mathbf{TM}_{\lambda-\Delta_\lambda}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}$*, where* $\Delta_\lambda = \ell^e(\log q + n)$*.*

Proof. Here, we omit the proof as it is technical and its main idea is analogous to the one presented in the proof of privacy. For the details see Appendix A.3. The high-level idea is to use reduction and apply Lemma 4 to bound the success probability.

References

1. Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings. pp. 36–54 (2009), http://dx.doi.org/10.1007/978-3-642-03356-8_3
2. Barak, B., Dodis, Y., Krawczyk, H., Pereira, O., Pietrzak, K., Standaert, F.X., Yu, Y.: Leftover Hash Lemma, revisited. IACR Cryptology ePrint Archive 2011, 88 (2011), http://dblp.uni-trier.de/db/journals/iacr/iacr2011.html#BarakDKPPSY11
3. Bennett, C., Brassard, G., Maurer, U.M.: Generalized privacy amplification. IEEE Transactions on Information Theory 41, 1915–1923 (1995)
4. Boyen, X., Dodis, Y., Katz, J., Ostrovsky, R., Smith, A.: Secure remote authentication using biometric data. In: Cramer, R. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 3494, pp. 147–163. Springer (2005), http://dblp.uni-trier.de/db/conf/eurocrypt/eurocrypt2005.html#BoyenDKOS05
5. Brakerski, Z., Kalai, Y.T.: A parallel repetition theorem for leakage resilience. In: Theory of Cryptography - 9th Theory of Cryptography Conference, TCC 2012, Taormina, Sicily, Italy, March 19-21, 2012. Proceedings. pp. 248–265 (2012), http://dx.doi.org/10.1007/978-3-642-28914-9_14
6. Christian Cachin, U.M.: Entropy Measures and Unconditional Security in Cryptography (1997)
7. Crescenzo, G.D., Lipton, R.J., Walfish, S.: Perfectly secure password protocols in the bounded retrieval model. In: TCC. pp. 225–244 (2006)
8. Damgård, I., Fehr, S., Renner, R., Salvail, L., Schaffner, C.: A tight high-order entropic quantum uncertainty relation with applications. In: Advances in Cryptology - CRYPTO 2007, 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2007, Proceedings. pp. 360–378 (2007), http://dx.doi.org/10.1007/978-3-540-74143-5_20
9. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. SIAM J. Comput. 38(1), 97–139 (2008)
10. Dodis, Y., Pietrzak, K., Wichs, D.: Key derivation without entropy waste. In: EUROCRYPT. pp. 93–110 (2014)

[3] id denotes the identity mapping

11. Dodis, Y., Yu, Y.: Overcoming weak expectations. In: TCC. pp. 1–22 (2013), http://dblp.uni-trier.de/db/conf/tcc/tcc2013.html#DodisY13
12. Dziembowski, S.: Intrusion-resilience via the bounded-storage model. In: Halevi, S., Rabin, T. (eds.) TCC. Lecture Notes in Computer Science, vol. 3876, pp. 207–224. Springer (2006)
13. Dziembowski, S., Kazana, T., Wichs, D.: Key-evolution schemes resilient to space-bounded leakage. In: Rogaway, P. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 335–353. Springer (2011)
14. Dziembowski, S., Pietrzak, K.: Leakage-resilient cryptography. In: FOCS. pp. 293–302. IEEE Computer Society (2008), http://dblp.uni-trier.de/db/conf/focs/focs2008.html#DziembowskiP08
15. Genkin, D., Shamir, A., Tromer, E.: Rsa key extraction via low-bandwidth acoustic cryptanalysis. In: CRYPTO (1). pp. 444–461 (2014)
16. Guruswami, V., Umans, C., Vadhan, S.P.: Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes. J. ACM 56(4) (2009)
17. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (Mar 1999), http://dx.doi.org/10.1137/S0097539793244708
18. Hoory, S., Linial, N., Wigderson, A., Overview, A.: Expander graphs and their applications. Bull. Amer. Math. Soc. (N.S 43, 439–561 (2006)
19. Kocher, P.: Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. pp. 104–113. CRYPTO '96, Springer-Verlag, London, UK, UK (1996), http://dl.acm.org/citation.cfm?id=646761.706156
20. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. Lecture Notes in Computer Science 1666, 388–397 (1999), citeseer.ist.psu.edu/kocher99differential.html
21. Merkle, R.C.: Secrecy, Authentication, and Public Key Systems. Ph.D. thesis, Stanford, CA, USA (1979), aAI8001972
22. Nisan, N., Ta-Shma, A.: Extracting randomness: A survey and new constructions. Journal of Computer and System Sciences 58(1), 148 – 173 (1999), http://www.sciencedirect.com/science/article/pii/S0022000097915464
23. Nisan, N., Zuckerman, D.: Randomness is linear in space. J. Comput. Syst. Sci. 52(1), 43–52 (1996), http://dx.doi.org/10.1006/jcss.1996.0004
24. Quisquater, J.J., Samyde, D.: Electromagnetic analysis (ema): Measures and countermeasures for smart cards. In: Attali, I., Jensen, T.P. (eds.) E-smart. Lecture Notes in Computer Science, vol. 2140, pp. 200–210. Springer (2001), http://dblp.uni-trier.de/db/conf/esmart/esmart2001.html#QuisquaterS01
25. Radhakrishnan, J., Ta-Shma, A.: Bounds for dispersers, extractors, and depth-two superconcentrators. Siam Journal on Discrete Mathematics 13, 2000 (2000)
26. Vadhan, S.: *Pseudorandomness. http://people.seas.harvard.edu/~salil/pseudorandomness/ (2012)*
27. *Wullschleger, J.: Oblivious-transfer amplification. In: Advances in Cryptology - EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings. pp. 555–572 (2007), http://dx.doi.org/10.1007/978-3-540-72540-4_32*
28. *Yao, Y., Li, Z.: Overcoming weak expectations via the Rényi entropy and the expanded computational entropy. In: Padró, C. (ed.) Information Theoretic Security - 7th International Conference, ICITS 2013, Singapore, November 28-30, 2013, Proceedings. Lecture Notes in Computer Science, vol. 8317, pp. 162–178. Springer (2013), http://dx.doi.org/10.1007/978-3-319-04268-8_10*

A Proofs of lemmata and theorems

A.1 Technical lemma

Here, we state a technical lemma that is a broad generalization of Lemma B.1 given in the work by Dziembowski, Kazana, and Wichs [13] and leads us to the proof of the main part of our work, i.e., Lemma 4 and its consequences. The statement may not look very interesting on its own: it is rather a wrapper for Corollary 2 and Corollary 3 stated in such a way that it will be directly used to prove Lemma 4. Similar idea was originally used in [13].

We start with a definition of a Guessing game which is tailored to be used in proof of Lemma 4 (intuitevely it describes exactly skills of adversary what we will deal with later).

Definition 8. *Let $(X, \mathcal{H})$ be random variables. A* Guessing game against adversary $\mathcal{A}$ *consists of the steps described in Fig. 5.*

Guessing **game for adversary $\mathcal{A}$.**

Input: random variables $(X, \mathcal{H})$, where $X = (X_1, \ldots, X_\ell)$ and $\mathcal{H} = (\mathcal{H}_{v_1}, \ldots, \mathcal{H}_{v_N})$ for some parameters ℓ, N and labels v_i. [a] Furthermore we declare k_1, k_2, leakage parameter λ_{Leak} and p such that X has min-entropy at least $p\ell \log N$.

Leakage phase:

1. $\mathcal{A}$ issues a leakage query $\text{Leak}(X_1, \ldots, X_\ell, \mathcal{H})$ of length λ_{Leak}.

First phase:

2. $\mathcal{A}$ adaptively queries $\mathcal{H}$ by submitting a label v and receiving $\mathcal{H}_v$.
3. $\mathcal{A}$ chooses a subset of indices $S_1 \subset [\ell]$ of size k_1 along with the guesses for all values $(X_j | j \in S_1)$.

Second phase:

4. $\mathcal{A}$ receives all values of $\{X_i | i \notin S_1\}$, i.e., all blocks that she did not try to guess.
5. $\mathcal{A}$ outputs a subset of labels $S_2 \subset \{v_i\}_{i=1..N}$ of size k_2 which were not previously (i.e., in the first phase) queried along with guesses for all values $(H_v | v \in S_2)$.

[a] The values of X_i should be considered as certain $\mathcal{H}$ labels.

Fig. 5. Definition of Guessing game

Lemma 5. *Let $(X_1, \ldots, X_\ell, \mathcal{H})$ be a random variable such that:*

1. *$(X_1, \ldots, X_\ell)$ and $\mathcal{H}$ are independent,*
2. *$\mathcal{H}$ is a vector of random independent $N = 2^{\delta n}$ blocks of length n,*
3. *each X_i is n bits long,*
4. *$\mathbf{H}_\infty(X_1, \ldots, X_\ell) = p\ell n$ for some $0 \leq p \leq 1$.*

Now let $\mathcal{A}$ be a randomized algorithm playing Guessing *game with $\lambda_{\text{Leak}} = \lambda n$. Then, the probability that $\mathcal{A}$ outputs all correct guesses (in both phases) is at most*

$$2^{-\varepsilon n} + 2^{-(\delta n - 1)} + 4\ell^2 \cdot 2^{-\beta n/4} + 2^{-n((p-\beta)\ell - \lambda - \varepsilon - 4\delta + k_2)}$$

if $k_1 > \ell - \lfloor \frac{\beta}{4}\ell \rfloor$ and $p\ell < N$, for any $\varepsilon > 0$ and $\beta < p$.

Proof. Let $X = (X_1, \ldots, X_\ell)$ and E be the list of all answers to q queries made by $\mathcal{A}$ in the first phase, and observe that $|E| = qn$. In this proof we deal with the distribution:

$$(X, \mathcal{H} | \mathrm{Leak}(X, \mathcal{H}) = l, E = e)$$

and apply the chain rule for it. Firstly, notice that by Lemma 1 with probability not less than $1 - 2^{-\varepsilon n}$ we have that:

$$\mathbf{H}_\infty(X, \mathcal{H} | \mathrm{Leak}(X, \mathcal{H}) = l, E = e) \geq p\ell n + Nn - \lambda_{\mathrm{Leak}} - |E| - \varepsilon n = n\left(p\ell + N - \lambda - q - \varepsilon\right).$$

By Corollary 1 for $\Delta = 2\delta$ we get that (with probability at least $1 - 2^{-(\Delta n - \log(p\ell + N - \lambda - q - \varepsilon))} \geq 1 - 2^{-(\Delta n - \log(N) - 1)} \geq 1 - 2^{-(\delta n - 1)}$ by the assumption $p\ell < N$):

$$\begin{aligned}\mathbf{H}_\infty(X | \mathrm{Leak}(X, \mathcal{H}) = l, E = e, \mathrm{Hint} = h) + \mathbf{H}_\infty(H | X, \mathrm{Leak}(X, \mathcal{H}) = l, E = e, \mathrm{Hint} = h) \geq \\ \geq n\left(p\ell + N - \lambda - q - \varepsilon - 4\delta\right)\end{aligned}$$

So, either:

$$\mathbf{H}_\infty(X | \mathrm{Leak}(X, \mathcal{H}) = l, E = e, \mathrm{Hint} = h) \geq \beta l n \tag{11}$$

or:

$$\mathbf{H}_\infty(\mathcal{H} | X, \mathrm{Leak}(X, \mathcal{H}) = l, E = e, \mathrm{Hint} = h) \geq n\left(p\ell + N - \lambda - q - \varepsilon - 4\delta - \beta\ell\right). \tag{12}$$

Now, as $k_1 > \ell - \lfloor \frac{\beta}{4}\ell \rfloor$ from Lemma 6 we know that the probability of all correct guesses in the first phase in case (11) is less then $4\ell^2 \cdot 2^{-\beta n/4}$. On the other hand, in case (12) we denote by $w_1, \ldots, w_{q+k_2}$ the labels of all queries conducted in the first phase together with the list of k_2 guesses from the second phase. Then, the probability of all correct guesses in the second phase is less then

$$2^{-\mathbf{H}_\infty(\mathcal{H}_{w_1}, \mathcal{H}_{w_2}, \ldots, \mathcal{H}_{w_{q+k_2}} | \mathrm{Leak}(X, \mathcal{H}) = l, E = e, \mathrm{Hint} = h)}.$$

After setting $E_{l,e,h}$ to be the event $(\mathrm{Leak}(X, \mathcal{H}) = l, E = e, \mathrm{Hint} = h)$ and $\widehat{\mathcal{H}}$ to be the complement of $\mathcal{H}_{w_1}, \mathcal{H}_{w_2}, \ldots, \mathcal{H}_{w_{q+k_2}}$ by Lemma 1 we see that:

$$\begin{aligned}\mathbf{H}_\infty(\mathcal{H}_{w_1}, \mathcal{H}_{w_2}, \ldots, \mathcal{H}_{w_{q+k_2}} | E_{l,e,h}) &\geq \widetilde{\mathbf{H}}_\infty(\mathcal{H}_{w_1}, \mathcal{H}_{w_2}, \ldots, \mathcal{H}_{w_{q+k_2}} | \widehat{\mathcal{H}}; E_{l,e,h}) \\ &= \widetilde{\mathbf{H}}_\infty(\mathcal{H} | \widehat{\mathcal{H}}; E_{l,e,h}) \geq \mathbf{H}_\infty(\mathcal{H} | E_{l,e,h}) - |\widehat{\mathcal{H}}| \\ &= \mathbf{H}_\infty(\mathcal{H} | E_{l,e,h}) - (N - q - k_2)n \\ &= n\left(p\ell + N - \lambda - q - \varepsilon - 4\delta - \beta\ell - N + q + k_2\right) \\ &= n\left(p\ell - \lambda - \varepsilon - 4\delta - \beta\ell + k_2\right).\end{aligned}$$

Therefore, by the union bound the final probability is bounded by:

$$2^{-\varepsilon n} + 2^{-(\delta n - 1)} + 4\ell^2 \cdot 2^{-\beta n/4} + 2^{-n((p-\beta)\ell - \lambda - \varepsilon - 4\delta + k_2)}.$$

Remark 8. Note that in the above reasoning we assumed that A learns $\mathrm{Leak}(X, \mathcal{H})$, E and Hint (cf. conditions in (11) and (12)). This might be confusing since in the definition of Guessing game we assumed that only $\mathrm{Leak}(X, \mathcal{H})$ and E are learned. However, any additional input may only increase the probability of winning in any game. Therefore, the statement is proven. We believe that such reasoning ("addi-

tional information may only help") is the major motivation for our definition of spoiling knowledge chain rule for min-entropy.

Lemma 6. *Let $X = (X_1, \ldots, X_l)$ be a sequence of (possibly dependent) random variables distributed over $\{0,1\}^n$. Now let $\mathcal{A}$ be a randomized algorithm that obtains a leakage $\mathrm{Leak}(X)$ as an input and consequently outputs a subset of $S \subset \{1, \ldots, l\}$ along with guesses for all values $\{X_i | i \in S\}$. Now, if $|S| > \ell - \lfloor \frac{\beta}{4}\ell \rfloor$ and $\widetilde{\mathbf{H}}_\infty(X|\mathrm{Leak}(X)) \geqslant \beta ln$, then the probability that A outputs all correct guesses is at most:*

$$4\ell^2 \cdot 2^{-\beta n/4}.$$

Proof. Here we just use Corollary 3: after learning $\mathrm{Leak}(X)$ and $\mathrm{Hint}(X)$ with probability at least $1 - 4\ell(\ell-1)2^{-\beta n/4}$ there are at least $\lfloor \frac{\beta}{4}\ell \rfloor$ blocks of min-entropy of at least $\beta n/4$. The algorithm $\mathcal{A}$ is supposed to guess more than $\ell - \lfloor \frac{\beta}{4}\ell \rfloor$ blocks so at least one block with min-entropy $\beta n/4$ is supposed to be guessed. This happens with probability at most $2^{-\beta n/4}$ and therefore by the union bound and the inequality $4\ell(\ell-1)+1 \leq 4\ell^2$ we obtain the claim. As described in remark above, here we also assume that A learns both $\mathrm{Leak}(X)$ and $\mathrm{Hint}(X)$.

A.2 Proof of one-wayness of Disperse

Here, we shall use the Guessing game defined in Appendix A.1. For a random oracle query $b \in \{0,1\}^{\ell^d n + \log \ell}$ we denote by $b[i]$ (for $i > 0$) an i-th n-elementary block of b and by $b[0]$ the $(\log \ell)$-elementary block i.e. $b = \langle b[0], b[1], \ldots, b[\ell^d] \rangle$.

Lemma 7 (One-wayness of Disperse). *Let $\mathcal{G}_\sigma$ be a ℓ^d-regular $(\ell^e, (1-\eta)\ell)$-right disperser and $D = (D_1, \ldots, D_\ell) \in \{0,1\}^{n\ell}$ be a random variable of min-entropy $p\ell n$. Then, for any ℓ sufficiently large the probability that an algorithm $\mathcal{A}(\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H})) \in \mathbf{TM}_{\lambda,q}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}$ makes at least ℓ^e different bad queries satisfies $\Pr(|\mathsf{indices}_\mathcal{A}| \geq \ell^e) = O(\ell^2 2^{-\beta n/4})$ for any β satisfying $p - \frac{\lambda}{\ell n} > \beta > 4\eta$ and $q = 2^{o(\ell^{1-e})n}$.*

Proof. Given an adversary $\mathcal{A}$ such that its associated list $\mathsf{indices}_\mathcal{A}$ is longer or equal to ℓ^e with probability ξ we construct a player $\mathcal{P}_\mathcal{A}$ in game $\mathsf{Guessing}_{\langle X_1,\ldots,X_\ell\rangle,\mathcal{H}}(p, k_1, k_2, \lambda_{\mathrm{Leak}})$ for

$$(X_1, \ldots, X_\ell) = (D_1, \ldots, D_\ell);$$
$$k_1 = (1-\eta)\ell > \ell - \lfloor \frac{\beta}{4}\ell \rfloor; \text{ satisfied for } \ell \text{ large enough by the assumption on } \beta$$
$$k_2 = \ell$$
$$\lambda_{\mathrm{Leak}} = \lambda + n\ell + \ell^e(\log q + \log \ell),$$

winning with probability ξ. Therefore, we conclude that

$$\xi < 2^{-\varepsilon n} + 2^{-(\ell^d n + \log \ell - 1)} + 4\ell^2 \cdot 2^{-\beta n/4} + 2^{-n\left((p-\beta-\frac{\lambda}{\ell n})\ell - \frac{\ell^e(\log q + \log \ell)}{n} - \varepsilon - 4(\ell^d + \frac{\log \ell}{n})\right)}$$

by Lemma 5 in Appendix A.1. The detailed construction of $\mathcal{P}_\mathcal{A}$ is described in Fig. 6.

Implementation of a player $\mathcal{P}_{\mathcal{A}}$.

Learning phase:

1. Player $\mathcal{P}_{\mathcal{A}}$ leaks ℓ values $(D'_1, \ldots, D'_\ell) = \left(\mathcal{H}(1, D_{\sigma_1^1} \ldots D_{\sigma_{\ell d}^1}), \ldots, \mathcal{H}(\ell, D_{\sigma_1^\ell} \ldots D_{\sigma_{\ell d}^\ell})\right)$ from a leakage oracle $\mathcal{O}$;
2. initializes adversary $\mathcal{A} \in \mathbf{TM}_{\lambda,q}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}$; feeds her with $D'_1, \ldots, D'_\ell$, then submits her to the leakage oracle $\mathcal{O}$[a] to get a hint $\mathrm{Leak}(D, \mathcal{H})$ containing:
   - the results of all leakage queries by $\mathcal{A}$;
   - a list $\mathsf{indices}'$ of at most ℓ^e first items of list $\mathsf{indices}_{\mathcal{A}}$

Game phase:

The adversary $\mathcal{A}$ is now executed with the following recipe for answering to leakage and oracle queries:

1. The answers to leakage queries are obtained using $\mathrm{Leak}(D, \mathcal{H})$.
2. The random oracle queries are obtained using $\mathcal{H}$ with an exception of elements $\mathcal{H}(j, D_{\sigma_1^j} \ldots D_{\sigma_{\ell d}^j})$ given in $\mathrm{Leak}(D, \mathcal{H})$.

First phase 1. Everytime $\mathcal{A}$ issues a bad oracle query $(i_k, D_{\sigma_1^{i_k}}, \ldots, D_{\sigma_{\ell d}^{i_k}})$ whose index k appears on the list $\mathsf{indices}'$ the player $\mathcal{P}_{\mathcal{A}}$ adds elements $D_{\sigma_j^{i_k}}$ to its list of guesses.

Second phase 1. $\mathcal{P}_{\mathcal{A}}$ guesses random oracle queries leaked in the first item of Learning phase.

[a] submitting $\mathcal{A}(D'_1, \ldots, D'_\ell)$ to $\mathcal{O}$ means that $\mathcal{O}$ gets a description of Turing Machine realising $\mathcal{A}$ along with input tape containing $D'_1, \ldots, D'_\ell$.

Fig. 6. Implementation of a player $\mathcal{P}_{\mathcal{A}}$

The first step is to show that a player $\mathcal{P}_{\mathcal{A}}$ follows the rules of the game:

$$\mathsf{Guessing}_{\langle D_1,\ldots,D_\ell\rangle,\mathcal{H}}\left(p, (1-\eta)\cdot\ell, \ell, \lambda + n\ell + \ell^e(\log q + \log \ell)\right).$$

For this sake, we show that:

- Length of hint $\lambda_{\mathrm{Leak}} = |\mathrm{Leak}(D_1, \ldots, D_\ell, \mathcal{H})|$ is not greater than leakage λ of the adversary $\mathcal{A}$ along with $n\ell$ bits needed to feed the $\mathcal{A}$ and $\ell^e(\log q + \log \ell)$ to handle bad queries. Hence $\lambda_{\mathrm{Leak}} = \lambda + n\ell + \ell^e(\log q + \log \ell)$.
- Rules of the game requires that $k_1 > \ell - \left\lfloor \frac{\beta\ell}{4} \right\rfloor$, which follows from the assumptions on β.

Now, we need to show that $\mathcal{P}_{\mathcal{A}}$: a) guesses at least k_1 elements in the first phase with probability ξ and b) guesses $k_2 = \ell$ elements in the second phase. The claim b) follows from directly from the definition of $\mathrm{Leak}(D, \mathcal{H})$. Namely, $\mathrm{Leak}(D, \mathcal{H})$ contains the values of $\mathcal{H}(i, D_{\sigma_1^i} \ldots D_{\sigma_{\ell d}^i})$ for $i = 1 \ldots \ell$ which are explicitly prohibited from being queried (Item 2 of the Game phase) and therefore can be guessed in the Second phase of operation of $\mathcal{P}_{\mathcal{A}}$. In order to prove a) we use the fact that every

bad query leads to the capability of guessing ℓ^d associated $D_{\sigma_j^i}$'s and apply the properties of disperser graphs. More precisely, by the assumptions on $\mathcal{A}$ the length of $\mathsf{indices}'$ is equal to ℓ^e with probability ξ. In this case, the neighbourhood of vertices labeled with $\mathsf{indices}' = \{i_1, \ldots, i_{\ell^e}\}$ consists of at least $(1-\eta)\ell$ elements (using basic property of disperser $\mathcal{G}_\sigma$) and therefore the elements $D_k^{i_j}$ for $j \in \{1, \ldots, \ell^e\}$ and $k \in \{1, \ldots, \ell^d\}$ can be guessed in the First phase of operation of $\mathcal{P}_\mathcal{A}$.

This leaves us with the proof that under our assumptions the value $2^{-\varepsilon n} + 2^{-(\ell^d n + \log \ell - 1)} + 4\ell^2 \cdot 2^{-\beta n/4} + 2^{-n\left(\left(p-\beta-\frac{\lambda}{\ell n}\right)\ell - \frac{\ell^e(\log q + \log \ell)}{n} - \varepsilon - 4(\ell^d + \frac{\log \ell}{n})\right)}$ is in fact equal to $O(4\ell^2 \cdot 2^{-\beta n/4})$. This follows from simple observations that: ε is arbitrary, $\ell^d > 1$, $\frac{\ell^e(\log q + \log \ell)}{n} - \varepsilon - 4(\ell^d + \frac{\log \ell}{n}) = o(\ell)$ (here we use $q = 2^{o(\ell^{1-e})n}$) and $p - \beta - \frac{\lambda}{\ell n} > 0$.

□

A.3 Proof of security of Disperse

Theorem 4 (Security). *Let* $\mathsf{Game} = (\mathcal{C}, \mathsf{id}, \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$[4] *be an* ε*-secure game based on randomness* $K \sim U_{n\ell}$ *for the class of adversaries* $\mathbf{TM}_\lambda^{\mathcal{O}(K)}$ *then for every* ℓ^d*-regular* $(\ell^e, (1-\eta)\ell)$*-disperser* $\mathcal{G}_\sigma$*,* β *satisfying* $p - \frac{\lambda}{\ell n} > \beta > 4\eta$ *and* $q = 2^{o(\ell^{1-e})n}$*, the game* $\mathsf{Game}_{\mathrm{Disk}} = (\mathcal{C}, \mathsf{Disperse}_{\mathcal{G}_\sigma}, \mathsf{Setup}_{\mathrm{Ch}}, \mathsf{Setup}_{\mathrm{Adv}}, \mathsf{Execute})$ *based on randomness* $(D, \mathcal{H})$ *of min-entropy* $p\ell n$ *is* $\varepsilon + O(\ell^2 \cdot 2^{\frac{-\beta n}{4}})$*-secure for adversaries in* $\mathbf{TM}_{\lambda-\Delta_\lambda}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}$*, where* $\Delta_\lambda = \ell^e(\log q + n)$.

Proof. The proof will be based on reduction. For the sake of contradiction, we assume that there exists (efficiently samplable) random distribution D of min-entropy $pn\ell$ and a $\mathsf{Game}_{\mathrm{Disk}}$-adversary $\mathcal{A} \in \mathbf{TM}_{\lambda-\Delta_\lambda,q}^{\mathcal{O}(D,\mathcal{H}),\mathcal{H}}$ such that:

$$\Pr_{D \leftarrow \mathcal{D}}(\mathsf{Game}_{\mathrm{Disk}}[\mathcal{A} \leftrightharpoons \mathcal{C}, \mathsf{key}' \leftarrow \mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H})] = \mathtt{Accept}) > \varepsilon' = \varepsilon + \Pr(|\mathsf{indices}_{\mathcal{A}_{\mathrm{Internal}}}| > \ell^e).$$

Using $\mathcal{A}$ as a component, we construct an adversary $\mathcal{A}' \in \mathbf{TM}_\lambda^{\mathcal{O}(K)}$ contradicting $(\varepsilon, \mathbf{TM}^{\mathcal{O}(K)})$-security of Game based on randomness K. Its description is given in Fig. 7. Note that we shall freely use the twisted random oracles defined in Definition 7.

To finish the proof we need the following claims.

Claim (Simulation). The execution of $\mathsf{G}' := \mathsf{Game}[\mathcal{A}' \leftrightharpoons \mathcal{C}, \mathsf{key} \leftarrow K]$ based on randomness K is in fact a simulation of $\mathsf{G} := \mathsf{Game}[\mathcal{A}_{\mathrm{Internal}} \leftrightharpoons \mathcal{C}, \mathsf{key}' \leftarrow \mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})]$ executed on mock randomness $(D, \mathcal{H})$.

Proof. Firstly, observe that the key K is equal to $\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})$ (by definition of $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}$) and therefore the input of $\mathcal{C}$ in G' is equal to $\mathsf{Setup}_{\mathrm{Ch}}(\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}))$ as in G. Moreover, all the messages send by $\mathcal{A}'$ are in fact produced by game G adversary $\mathcal{A}_{\mathrm{Internal}}$ and therefore the only

[4] id denotes the identity mapping

difference is that the leakage and random oracle queries of $\mathcal{A}_{\text{Internal}}$ are not processed honestly but simulated by means of leakage of $\mathcal{A}'$ described in steps (3a) and (3b) in Fig. 7.

Claim (Simulation's correctness). The simulation above is faithful (i.e., $\mathcal{A}'$ works the same as corresponding $\mathcal{A}_{\text{Internal}}$) unless $\perp$ is returned in part (3c) of simulation. This occurs with probability $\xi = \Pr(|\mathsf{indices}| > \ell^e)$ and therefore

$$\Pr(\mathcal{A}_{\text{Internal}} \text{ is faithfully simulated}) = 1 - \Pr(|\mathsf{indices}| > \ell^e) = 1 - \xi.$$

Proof. The first part concerning simulation correctness is clear from the construction. More precisely, the difference between messages of honest $\mathcal{A}_{\text{Internal}}$ and $\mathcal{A}'$ occurs only if the restriction λ on the size of leakage (see (3c)) intervenes. Therefore we are left to prove that $\Pr(\perp \text{ is returned in (3c)}) = \xi$. This follows directly by applying Lemma 4 for $\mathcal{A}_{\text{Internal}}$.

Claim (Leakage bound). The total leakage of $\mathcal{A}'$ during the execution of $\mathsf{Game}(\mathcal{A}' \leftrightharpoons \mathcal{C}, \mathtt{key} \leftarrow K)$ does not exceed λ bits and consequently $\mathcal{A}'$ belongs to $\mathbf{TM}_{\lambda}^{\mathcal{O}(K)}$.

Proof. Clear, as we have explicitly bounded the leakage inside $\mathcal{A}'$ (see (3c) in Fig. 7).

All above claims prove that $\mathcal{A}'$ is an efficient adversary for Game which succeeds with probability:

$$\begin{aligned}
\Pr(\mathsf{G}' = \mathtt{Accept}) &\geqslant \Pr(G = \mathtt{Accept} \text{ and } \mathcal{A}_{\text{Internal}} \text{ is correctly simulated}) && (13)\\
&\geqslant \Pr(G = \mathtt{Accept}) + \Pr(\mathcal{A}_{\text{Internal}} \text{ is correctly simulated}) - 1 && (14)\\
&= \varepsilon' + (1 - \xi) - 1 = \varepsilon' - \xi > \varepsilon, && (15)
\end{aligned}$$

where in line (13) we used (Simulation) claim and in (15) we used the fact that the distributions $(D, \mathcal{H})$ and $(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})$ are the same and therefore

$$\Pr(G = \mathtt{Accept}) = \Pr(\mathsf{Game}[\mathcal{A} \leftrightharpoons \mathcal{C}, \mathtt{key} \leftarrow \mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H})] = \mathtt{Accept}) = \varepsilon',$$

and moreover that $\Pr(\mathcal{A}_{\text{Internal}} \text{ is correctly simulated}) = 1 - \xi$ by Claim (Simulation's correctness).

This contradicts the $(\varepsilon, \mathbf{TM}^{\mathcal{O}(K)})$-security of Game and consequently gives a proof of the theorem as by Lemma 7 the probability $\Pr(|\mathsf{indices}_{\mathcal{A}_{\text{Internal}}}| > \ell^e) = O(\ell^2 \cdot 2^{\frac{-\beta n}{4}})$.

B Disperser graphs

Here, we present some structure theorems concerning so-called disperser graphs, which allows us to amplify privacy and security properties of the above constructions. We do not claim originality of the upcoming considerations. Similar arguments appear for example in [18, 26]. Despite this, we were not able to find the results of Corollary 4 in literature. Let $\mathcal{G} = (V, E)$ be an undirected graph with the set of vertices V and edges E. For a subset $S \subset V$ by $N(S)$ we denote $\{w \in V : \exists_{s \in S}(s, w) \in E\}$, i.e. the set of all neighbours of S. We say that a bipartite graph $\mathcal{G} = (V^0 \sqcup V^1, E)$ is *right d-regular* if all vertices in V^1 have the same degree d.

Definition 9 ((k,ε)-extractor). *We say that a function* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is an* (k,ε)*-extractor if for any random variable* X *of min-entropy* $\mathbf{H}_\infty(X) \geq k$ *and* $S \sim U_d$ *we have* $\Delta(\mathrm{Ext}(X,S), U_m) \leq \varepsilon$.

Definition 10. *A bipartite graph* $\mathcal{G} = (V^0 \sqcup V^1, E)$ *is a* right (K,L)-disperser *if for every set* $S \subset V^1$ *such that* $|S| = K$ *the neighbourhood* $N(S)$ *satisfies*

$$|N(S)| \geqslant L,$$

i.e. the sets of size K *expands into sets of size at least* L.

We shall use the following simple lemma which describes expansion properties of bipartite graphs.

Lemma 8. *Let* $\mathcal{G} = (V^0 \sqcup V^1, E)$ *be a bipartite right* (K,L)*-disperser. Then for every set* $S \subset V^1$ *such that* $|S| \geqslant K$ *and every subset* $T \subset V^0$ *of size* $|T| < L$ *the set* $N(S)$ *is not contained in* T.

Proof. This is a direct counting. Take $S' \subset S$ of size equal to K. The size of $N(S)$ then satisfies:

$$|N(S)| \geqslant |N(S')| = L.$$

which finishes the proof.

In order to instantiate the above considerations, we need the following correspondence between dispersers and randomness extractors. Observe that every function $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ corresponds to a right 2^m-regular bipartite graph $\mathcal{G}_f = (V_f, E_f)$ defined by:

$$V_f = V_f^0 \sqcup V_f^1 \text{ for } V_f^0 = V_1^f = \{0,1\}^n \tag{16}$$

$$E_f = \left\{(v_0, v_1) : \exists_{e \in \{0,1\}^m} f(v_1, e) = v_0\right\}. \tag{17}$$

It turns out that randomness dispersing properties of f are closely related to the vertex expansion of $\mathcal{G}_f$. Namely, the following theorem holds.

Proposition 2. *(see [26], Proposition 6.20) Let* $D : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}^n$ *be a function such that for any random variable* $X \in \{0,1\}^n$ *of min-entropy* $H_\infty(X) \geqslant \delta n$ *and* E *distributed uniformly on* $\{0,1\}^m$*, the statistical distance* $\Delta(D(X,E), U_n)$ *is less or equal to* ε*. Then for every* S*, a subset of right side of a bipartite graph* $\mathcal{G}_D = (V_D^0 \sqcup V_D^1, E_D)$ *of size* $2^{\delta n}$ *the neighbourhood* $N(S)$ *satisfies*

$$|N(S)| \geqslant (1-\varepsilon) \cdot 2^n,$$

i.e. the graph $\mathcal{G}_D$ *is a right* 2^m*-regular* $\left(2^{\delta n}, (1-\varepsilon)\cdot 2^n\right)$*-disperser.*

Proof. Take S a subset of $\{0,1\}^n$ of size $2^{\delta n}$, X_S a random variable distributed uniformly on S and E distributed uniformly on $\{0,1\}^m$. Observe that $H_\infty(X_S) \geqslant \delta n$ and therefore, by assumptions concerning D, the inequality $\Delta(D(X_S,E), U_n) \leqslant \varepsilon$ holds. Moreover, we see that $\Pr(D(X_S,E) = s) \neq 0$ exactly for $s \in N(S)$, so $\Delta(D(X,S), U_n) \geqslant (2^n - |N(S)|) \cdot \frac{1}{2^n}$. Combining these two inequalities we obtain

$$\varepsilon \geqslant \Delta(D(X_S,E), U_n) \geqslant (2^n - |N(S)|) \cdot \frac{1}{2^n},$$

which is equivalent to the proposition.

The existence of an efficiently computable function Ext (we require efficient computability in order to instantiate $\mathcal{G}_{\mathrm{Ext}}$ effectively), satisfying the assumptions of Proposition 2 follows from the result proven in [16]:

Theorem 5 (see [16], Theorem 1.5). *For every constant $\alpha > 0$ and all positive integers n, k and all $\varepsilon > 0$, there is an explicit construction of a (k, ε)-extractor $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^s \to \{0,1\}^m$ with $s = O(\log n + \log(1/\varepsilon))$ and $m \geqslant (1-\alpha)k$.*

More precisely, for any $\eta > 0$ we take an $((1+\eta), \varepsilon)$-extractor $\mathrm{Ext} : \{0,1\}^{(1+\gamma+\eta)n} \times \{0,1\}^s \to \{0,1\}^n$ which corresponds to the choice $\alpha = \frac{\eta}{1+\eta}$ in above Theorem 5, and interpret it as a function $\widetilde{\mathrm{Ext}} : \{0,1\}^n \times \{0,1\}^{(\gamma+\eta)n+s} \to \{0,1\}^n$. Then, taking $X \in \{0,1\}^n$ of min-entropy $\mathbf{H}_\infty(X) \geq (1-\gamma)n$ we see that $\Delta(\widetilde{\mathrm{Ext}}(X, U_{(\gamma+\eta)n+s}), U_n) \leq \varepsilon$ and therefore we may apply Proposition 2 to obtain a $2^{(\gamma+\eta)n+s}$-regular $(2^{(1-\gamma)n}, (1-\varepsilon)\cdot 2^n)$-disperser. As η might be chosen arbitrarily and $s = O(\log n + \log(1/\varepsilon))$ we obtain:

Corollary 4. *For any $\varepsilon > 0$ and $c > 0$, there exists $n_c \in \mathbb{N}$ such that for $n > n_c$ and (d, k) such that $d \cdot k \geq 2^{(1+c)n}$, there exists an effectively computable bipartite right d-regular graph which is a right $(k, (1-\varepsilon)\cdot 2^n)$-disperser. In particular, for any $\alpha > \frac{1}{2}$ and $\ell = 2^n$ sufficiently large there exists a bipartite right ℓ^α-regular $(\sqrt{\ell}, (1-\varepsilon)\cdot\ell)$-disperser.*

Proof. Apply Proposition 2 and the above discussion after Theorem 5.

C Proofs of auxiliary facts

Proof (Proof of Proposition 1). Let u_γ be a shorthand for $\lfloor \frac{\beta-\gamma}{1-\gamma}\ell \rfloor$. Assume the opposite, i.e., there exist $\ell - u_\gamma$ numbers x_i smaller than γn. Then,

$$x_1 + \cdots + x_\ell < u_\gamma n + (\ell - u_\gamma)\gamma n = \big((1-v)u_\gamma + \gamma\ell\big)\cdot n \leq \big((1-\gamma)\frac{\beta-\gamma}{1-\gamma} + \gamma\big)\cdot \ell n = \beta\ell n\,,$$

which contradicts the fact that all x_i sum up to $\beta\ell n$.

Implementation of $\mathcal{A}'$.

Input: The adversarial data $\mathsf{Setup}_{\mathrm{Adv}}(K)$ on K sampled from $\sim U_{n\ell}$ (cf. step (1) in Fig. 1).

Setup phase:

1. A mock random variable D is sampled from the distribution $\mathcal{D}$. An efficient data structure **OracleQueryList** for the on-the-fly storage of random oracle $\mathcal{H}$ queries is prepared.
2. An internal version $\mathcal{A}_{\mathrm{Internal}}$ of $\mathsf{Game}_{\mathrm{Disk}}$ adversary $\mathcal{A}$ is initialized and given

$$\mathsf{Setup}_{\mathrm{Adv}}(K) = \mathsf{Setup}_{\mathrm{Adv}}(\mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\}))$$

as input (as in $\mathsf{Game}[\mathcal{A}_{\mathrm{Internal}} \leftrightharpoons \mathcal{C}, \mathtt{key}' \leftarrow \mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})]$).

Execution phase:

3. Every time $\mathcal{A}'$ is provided with a new message $\mathtt{msg}_{\mathcal{A}'}$ (including the one initialized with the 0-th message $\perp$), she performs the following steps:
   a) performs getIndices() leakage:

> **getIndices() leakage**
>
> **Result** The list $\mathsf{indices}$ containing pairs (i, v_i) of an index i and a result v_i of $\mathcal{A}_{\mathrm{Internal}}$'s bad random oracle query (cf. Definition 6) conducted during its operation in $\mathsf{Game}[\mathcal{A}_{\mathrm{Internal}} \leftrightharpoons \mathcal{C}, \mathtt{key}' \leftarrow \mathsf{Disperse}_{\mathcal{G}_\sigma}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})]$ when provided with $\mathtt{msg}_{\mathcal{A}'}$ before sending the next message.
> **Description of the function** Just simulate the behaviour of $\mathcal{A}_{\mathrm{Internal}}$ and check whether random oracle queries are bad or not.
> **Complexity** Leakage: $|\mathsf{indices}| \cdot (\log q + n)$, time: the same as $\mathcal{A}_{\mathrm{Internal}}$ operation

   b) simulates the behaviour of $\mathcal{A}_{\mathrm{Internal}}$ with the following recipe for answering leakage and random oracle queries:

> **How to reply to leakage and random oracle queries?**
>
> **Random oracle queries** If the index i appears in one of the pairs in the list $\mathsf{indices}$ leaked above then return v_i otherwise look up **OracleQueryList** and answer with a value from there or a random element drawn from U_n. In any case, add the whole query to **OracleQueryList**.
> **Leakage queries** We answer a leakage oracle query f described by a circuit containing $\mathcal{O}(D, \mathcal{H})$ queries by the same circuit containing $\mathcal{O}(D, \mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})$ instead. Note that in order to substitute all bad queries of $\mathcal{H}$ by $\mathcal{H}\{D \xrightarrow{\mathcal{G}_\sigma} K\})$, we just need to access D, $\mathcal{H}$ and $\mathcal{O}(K)$ which are all given to $\mathcal{A}'$.
> **Complexity** Leakage: same as $\mathcal{A}_{\mathrm{Internal}}$'s, time: same as $\mathcal{A}_{\mathrm{Internal}}$'s up to time necessary for **OracleQueryList** look ups. Bounded by $\lambda - \Delta_\lambda$.

   c) if the total leakage equal to $\lambda - \Delta_\lambda + |\mathsf{indices}| \cdot (\log q + n)$ exceeds λ then terminate with $\perp$;
   d) returns the message prepared by $\mathcal{A}_{\mathrm{Internal}}$.

Fig. 7. Implementation of $\mathcal{A}'$